``````

# create_docs.sh

``````
Shell script that generates class documentations.
This is not very clean, but works for now.
Generated documentations:
./docs/java/   - Java HTML documentation
./docs/js/   - JavaScript HTML documentation
./docs/moveatis_java_class_documentation.pdf   - Java PDF documentation
./docs/moveatis_js_class_documentation.html   - JavaScript one page HTML
documentation

mkdir docs
mkdir docs/js
mkdir docs/js-onepage
mkdir docs/java-tex

jsdoc_path="../node_modules/jsdoc"
js_onepage_path="../node_modules/jsdoc-one-page"
js_out_path="docs/js"
js_onepage_out_path="docs/js-onepage"
js_src_path="src/main/webapp/META-INF/resources/js"
js_class_doc_file="docs/moveatis_js_class_documentation.html"
js_front_page="docs/js-front-page.md"

texdoclet_path="../TeXDoclet.jar"
tex_init_file="docs/tex_init.tex"
tex_out_path="docs/java-tex"
tex_out_file="moveatis_java_class_documentation.tex"
java_out_path="docs/java"
java_class_doc_file="moveatis_java_class_documentation.pdf"
java_src_path="src/main/java"

JavaScript documentation front page

echo "
Moveatis JavaScript Class Documentation
Software version 2.0.0

Documentation version 1.0.0

Moveatis is a web application designed to help the analysis of teaching situations
by means of systematic observation. It was developed for the Department of Sport
Pedagogy at University of Jyväskylä. Moveatis was written in Java and JavaScript
programming languages. The JavaScript classes are documented here and the Java
classes are described in a separate class documentation.

(c) Copyright 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio and
Ilari Paananen.
(c) Copyright 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala and Karoliina
Lappalainen.
" > $js_front_page

JavaScript documentation
``````
``````

```
$jsdoc_path -d $js_out_path $(find_cyg $js_src_path -name *.js) $js_front_page
$jsdoc_path -t $js_onepage_path -d $js_onepage_out_path $(find_cyg $js_src_path
-name *.js) $js_front_page

echo "<!DOCTYPE html>
<html><head>
    <meta charset='utf-8'/>
    <title>Moveatis JavaScript Class Documentation</title>
    <style>
        .back-to-top { display: none; }
        td { padding: 0 0.5em; }
        footer { padding-top: 2em; }
    </style>
</head><body>" > $js_class_doc_file

cat $js_onepage_out_path/api_content_only.html >> $js_class_doc_file

grep -A 2 "<footer>" $js_onepage_out_path/index.html >> $js_class_doc_file

echo "<script>
function appendToc(toc, heading, items) {
    var h3 = document.createElement('h3');
    h3.appendChild(document.createTextNode(heading));
    toc.appendChild(h3);
    var ul = document.createElement('ul');
    for (var i = 0; i < items.length; i++) {
        var li = document.createElement('li');
        li.appendChild(document.createTextNode(items[i]));
        ul.appendChild(li);
    }
    toc.appendChild(ul);
}
(function() {
    var sections = document.getElementsByTagName('section');
    var classes = [];
    var modules = [];
    for (var i = 0; i < sections.length; i++) {
        var s = sections[i];
        var id = s.getAttribute('id');
        if (id === null) continue;
        var name;
        if (id.startsWith('class')) {
            name = id.slice(id.indexOf('~') + 1);
            classes.push(name);
            name = 'Class: ' + name;
        } else if (id.startsWith('module')) {
            name = id.slice(id.indexOf(':') + 1);
            modules.push(name);
            name = 'Module: ' + name;
        }
        var h1 = document.createElement('h1');
        h1.appendChild(document.createTextNode(name));
        s.parentElement.insertBefore(h1, s);
    }
    var toc = document.createElement('section');
```

```
    var h2 = document.createElement('h2');
    h2.appendChild(document.createTextNode('Table of Contents'));
    toc.appendChild(h2);
    appendToc(toc, 'Classes', classes);
    appendToc(toc, 'Modules', modules);
    sections[0].parentElement.insertBefore(toc, sections[0].nextSibling);

    var headers = document.getElementsByTagName('header');
    while (headers.length) {
        headers[0].remove();
        headers = document.getElementsByTagName('header');
    }
})();
</script></body></html>" >> $js_class_doc_file

# Java documentation

javadoc -d $java_out_path -sourcepath $java_src_path -subpackages com
mvn generate-sources javadoc:javadoc
cp -r target/site/apidocs/ docs/java/

echo "\\usepackage[utf8]{inputenc}" > $tex_init_file

javadoc -docletpath $texdoclet_path -doclet org.stfm.texdoclet.TeXDoclet \
-texinit $tex_init_file \
-tree -output $tex_out_path/$tex_out_file \
-title "Moveatis Java Class Documentation\\\\Software version 2.0.0" \
-subtitle "Version 1.0.0" \
-author "Jarmo Juuj\\""\"arvi\\\\Sami Kallio\\\\Kai Korhonen\\\\Juha
Moisio\\\\Ilari Paananen\\\\Visa Nyk\\""\"anen\\\\Petra Puumala\\\\ Karoliina
Lappalainen\\\\Tuomas Moisio" \
-nosummaries \
-sourcepath $java_src_path -subpackages com

cd $tex_out_path

pdflatex $tex_out_file
pdflatex $tex_out_file

cp $java_class_doc_file ../


``````

moveatis_java_class_documentation.log

``````
This is pdfTeX, Version 3.14159265-2.6-1.40.19 (MiKTeX 2.9.6930 64-bit)
(preloaded format=pdflatex 2019.2.3)  1 MAY 2019 20:19
entering extended mode
**moveatis_java_class_documentation.tex


(moveatis_java_class_documentation.tex
LaTeX2e <2018-12-01>
)
```

```
*X

! LaTeX Error: Missing \begin{document}.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
 ...

<*> X

? X

Here is how much of TeX's memory you used:
 8 strings out of 492946
 418 string characters out of 3135502
 57009 words of memory out of 3000000
 4004 multiletter control sequences out of 15000+200000
 3640 words of font info for 14 fonts, out of 3000000 for 9000
 1141 hyphenation exceptions out of 8191
 5i,0n,4p,6b,14s stack positions out of 5000i,500n,10000p,200000b,50000s
No pages of output.
PDF statistics:
 0 PDF objects out of 1000 (max. 8388607)
 0 named destinations out of 1000 (max. 500000)
 1 words of extra memory for PDF output out of 10000 (max. 10000000)


``````
```

```
nbactions-Moveatis.xml

``````
<?xml version="1.0" encoding="UTF-8"?>
<actions>
    <action>
        <actionName>run</actionName>
        <packagings>
            <packaging>war</packaging>
            <packaging>ear</packaging>
            <packaging>ejb</packaging>
        </packagings>
        <goals>
            <goal>package</goal>
        </goals>
    </action>
</actions>

``````
```

```
src/main/java/com/moveatis/abstracts/AbstractObservationEntity.java

``````
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
```

```
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.abstracts;

import javax.persistence.ManyToOne;
import javax.persistence.MappedSuperclass;

import com.moveatis.event.EventEntity;
import com.moveatis.observation.ObservationEntity;
import com.moveatis.user.AbstractUser;

/**
 * @author Visa Nykänen Superclass for the common features of feedback analyses
 *         and observations
 */
@MappedSuperclass
public abstract class AbstractObservationEntity extends BaseEntity {

    /**
     * The user doing the observing, can be public user or logged in user
     */
    @ManyToOne
    private AbstractUser observer;

    /**
     * The event for which the analysis or observation is made
     */
    @ManyToOne
```

```
private EventEntity event;

/**
 * The duration of the analysis or observation event
 */
private long duration;

/**
 * The description of the analysis or observation event
 */
private String description;

@Override
public Long getId() {
    return id;
}

@Override
public void setId(Long id) {
    this.id = id;
}

public AbstractUser getObserver() {
    return observer;
}

public void setObserver(AbstractUser observer) {
    this.observer = observer;
}

public EventEntity getEvent() {
    return event;
}

public void setEvent(EventEntity event) {
    this.event = event;
}

public long getDuration() {
    return duration;
}

public void setDuration(long duration) {
    this.duration = duration;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

@Override
public int hashCode() {
```

```
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof ObservationEntity)) {
            return false;
        }
        ObservationEntity other = (ObservationEntity) object;
        return !((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id)));
    }

    @Override
    public String toString() {
        return "com.moveatis.observation.ObservationEntity[ id=" + id + " ]";
    }

}
```

```
# src/main/java/com/moveatis/abstracts/AbstractRecordEntity.java
```

```
package com.moveatis.abstracts;

import javax.persistence.MappedSuperclass;

import com.moveatis.records.RecordEntity;

/**
 * @author Visa Nykänen, Sami Kallio Superclass for the common features of
 *         records in observations and feedback analysis
 */
@MappedSuperclass
public abstract class AbstractRecordEntity extends BaseEntity {

    /**
     * The time at which the record takes place
     */
    private Long startTime;

    /**
     * Written description of the recorded event
     */
    private String comment;

    public Long getStartTime() {
        return startTime;
    }

    public void setStartTime(Long startTime) {
        this.startTime = startTime;
```

```
    }

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        if (!(object instanceof RecordEntity)) {
            return false;
        }
        RecordEntity other = (RecordEntity) object;
        return !((this.id == null && other.id != null) || (this.id != null &&
!this.id.equals(other.id)));
    }

    @Override
    public String toString() {
        return "com.moveatis.records.RecordEntity[ id=" + id + " ]";
    }

}
```

# src/main/java/com/moveatis/feedbackanalysis/FeedbackAnalysisBean.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
```

```
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.feedbackanalysis;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

import javax.ejb.EJB;
import javax.ejb.Stateful;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.TypedQuery;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.moveatis.abstracts.AbstractBean;
import com.moveatis.interfaces.Event;
import com.moveatis.interfaces.FeedbackAnalysisRecord;
import com.moveatis.interfaces.FeedbackAnalysis;
import com.moveatis.observation.ObservationBean;
import com.moveatis.records.FeedbackAnalysisRecordEntity;
import com.moveatis.session.SessionBean;
import com.moveatis.user.AbstractUser;

/**
 * Manages the database connection for the feedback analyses
 *
 * @author Visa Nykänen
 */
@Stateful
public class FeedbackAnalysisBean extends AbstractBean<FeedbackAnalysisEntity>
        implements FeedbackAnalysis, Serializable {
    private static final Logger LOGGER =
LoggerFactory.getLogger(ObservationBean.class);

    private static final long serialVersionUID = 1L;

    @Inject
```

```
    private SessionBean sessionBean;

    @EJB
    private Event eventEJB;

    private FeedbackAnalysisEntity feedbackAnalysisEntity;

    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
    private EntityManager em;

    @Inject
    private FeedbackAnalysisRecord feedbackAnalysisRecordEJB;

    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public FeedbackAnalysisBean() {
        super(FeedbackAnalysisEntity.class);
    }

    /**
     * Finds and returns all observations for the specific user.
     *
     * @param observer
     *            The user, whose analyses are to be searched.
     * @return A list of the analyses for the user.
     */
    @Override
    public List<FeedbackAnalysisEntity> findAllByObserver(AbstractUser
observer) {
        TypedQuery<FeedbackAnalysisEntity> query =
em.createNamedQuery("findFeedbackAnalysesByObserver",
                FeedbackAnalysisEntity.class);
        query.setParameter("observer", observer);
        return query.getResultList();
    }

    /**
     * Finds the analyses for the user, which have no event attached to them.
     *
     * @param observer
     *            The user, whose analyses are to be searched.
     * @return A list of the analyses.
     */
    @Override
    public List<FeedbackAnalysisEntity> findWithoutEvent(AbstractUser observer)
{
        TypedQuery<FeedbackAnalysisEntity> query =
em.createNamedQuery("findFeedbackAnalysesWithoutEvent",
                FeedbackAnalysisEntity.class);
        query.setParameter("observer", observer);
        return query.getResultList();
    }
```

```
    /**
     * Finds the analyzatinos that are made for events that the specified user
does
     * not own.
     *
     * @param observer
     *            The user, whose analyses are to be searched.
     * @return A list of the analyses.
     */
    @Override
    public List<FeedbackAnalysisEntity> findByEventsNotOwned(AbstractUser
observer) {
        TypedQuery<FeedbackAnalysisEntity> query =
em.createNamedQuery("findFeedbackAnalysesByEventsNotOwned",
                FeedbackAnalysisEntity.class);
        query.setParameter("observer", observer);
        return query.getResultList();
    }

    /**
     * Persists the observations to the database.
     *
     * @param feedbackAnalysis
     *            The observatio entity to be persisted.
     */
    @Override
    public void create(FeedbackAnalysisEntity feedbackAnalysis) {
        super.create(feedbackAnalysis);
    }

    @Override
    public void edit(FeedbackAnalysisEntity feedbackAnalysis) {
        super.edit(feedbackAnalysis);
    }

    /**
     * Finds a list of the records for the observation with the given id.
     *
     * @param id
     *            The id of the analysis.
     * @return A list of the records.
     */
    @Override
    public List<FeedbackAnalysisRecordEntity> findRecords(Object id) {
        feedbackAnalysisEntity = em.find(FeedbackAnalysisEntity.class, id);
        if (feedbackAnalysisEntity != null) {
            return feedbackAnalysisEntity.getRecords();
        }
        return new ArrayList<>(); // return empty list
    }

    /**
     * Removes the analysis and also removes the analysis from the event it was
     * associated with.
     *
     * @param feedbackAnalysisEntity
```

```
     *            The analysis to be removed.
     */
    @Override
    public void remove(FeedbackAnalysisEntity feedbackAnalysisEntity) {
        super.remove(feedbackAnalysisEntity);
        eventEJB.removeFeedbackAnalysis(feedbackAnalysisEntity);
        feedbackAnalysisEntity.setEvent(null);
        super.edit(feedbackAnalysisEntity);
    }

    @Override
    public void removeRecordFromAnalysis(FeedbackAnalysisEntity
feedbackAnalysis, FeedbackAnalysisRecordEntity record) {
        List<FeedbackAnalysisRecordEntity> records =
feedbackAnalysis.getRecords();
        records.remove(record);
        record.setFeedbackAnalysis(null);
        feedbackAnalysis.setRecords(records);
        feedbackAnalysisRecordEJB.remove(record);
        for (FeedbackAnalysisRecordEntity rec : feedbackAnalysis.getRecords())
{
            if (rec.getId() == null)
                feedbackAnalysisRecordEJB.create(rec);
            else
                feedbackAnalysisRecordEJB.edit(rec);
        }
        super.edit(feedbackAnalysis);
    }

    /**
     * Permanently removes the analysis, which the user did not set to be saved
into
     * the database.
     *
     * @param feedbackAnalysisEntity
     *            The analysis to be removed.
     */
    @Override
    public void removeUnsavedObservation(FeedbackAnalysisEntity
feedbackAnalysisEntity) {
        em.remove(em.merge(feedbackAnalysisEntity));
    }
}

``````

src/main/java/com/moveatis/feedbackanalysis/FeedbackAnalysisEntity.java

``````
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
```

```
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.feedbackanalysis;

import static javax.persistence.CascadeType.ALL;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;

import com.moveatis.abstracts.AbstractObservationEntity;
import com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategoryEntity;
import
com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategorySetEntity;
import com.moveatis.records.FeedbackAnalysisRecordEntity;

/**
 * The entity for the feedback analysis, corresponds to the
 * feedbackanalysis-table in the database
 *
 * @author Visa Nykänen
 */
@Table(name = "FEEDBACKANALYSIS")
@NamedQueries({
        @NamedQuery(name = "findFeedbackAnalysesByObserver", query = "SELECT
```

```
analysis FROM FeedbackAnalysisEntity analysis WHERE analysis.observer=:observer
AND analysis.removed is null"),
        @NamedQuery(name = "findFeedbackAnalysesWithoutEvent", query = "SELECT
analysis FROM FeedbackAnalysisEntity analysis WHERE analysis.observer=:observer
AND analysis.event is null AND analysis.removed is null"),
        @NamedQuery(name = "findFeedbackAnalysesByEventsNotOwned", query =
"SELECT analysis FROM FeedbackAnalysisEntity analysis WHERE
analysis.observer=:observer AND analysis.event.creator<>:observer AND
analysis.removed is null") })
@Entity
public class FeedbackAnalysisEntity extends AbstractObservationEntity {

    @OneToMany(mappedBy = "feedbackAnalysis", fetch = FetchType.LAZY, cascade =
ALL)
    private List<FeedbackAnalysisRecordEntity> records;

    private String analysisName;

    private String targetOfAnalysis;

    public List<FeedbackAnalysisRecordEntity> getRecords() {
        return records;
    }

    public void setRecords(List<FeedbackAnalysisRecordEntity> records) {
        this.records = records;
    }

    public void addRecord(FeedbackAnalysisRecordEntity record) {
        if (this.getRecords() == null) {
            this.records = new ArrayList<>();
        }
        getRecords().add(record);
        record.setFeedbackAnalysis(this);
    }

    /**
     * The feedbackanalysiscategorysets for the feedback analysis aren't saved
so
     * they have to be fetched by getting the categorysets to which all the selected
     * categories belong to This means that if an analysis has a categoryset that
is
     * not used in the analysis, it won't be accessible when loading the analysis
     * later.
     *
     * @return the feedbackanalysiscategorysets used by the analysis
     */
    public List<FeedbackAnalysisCategorySetEntity>
getFeedbackAnalysisCategorySets() {
        List<FeedbackAnalysisCategorySetEntity>
feedbackAnalysisCategorySetsInUse = new
ArrayList<FeedbackAnalysisCategorySetEntity>();
        for (FeedbackAnalysisRecordEntity record : records) {
            for (FeedbackAnalysisCategoryEntity cat :
record.getSelectedCategories()) {
                boolean contained = false;
```

```
                for (FeedbackAnalysisCategorySetEntity facs :
feedbackAnalysisCategorySetsInUse) {
                    contained =
facs.getId().equals(cat.getCategorySet().getId());
                    if (contained)
                        break;
                }
                if (!contained)

    feedbackAnalysisCategorySetsInUse.add(cat.getCategorySet());
            }
        }
        return feedbackAnalysisCategorySetsInUse;
    }

    public String getAnalysisName() {
        return analysisName;
    }

    public void setAnalysisName(String name) {
        this.analysisName = name;
    }

    public String getTargetOfAnalysis() {
        return targetOfAnalysis;
    }

    public void setTargetOfAnalysis(String target) {
        this.targetOfAnalysis = target;
    }
}
```

# src/main/java/com/moveatis/helpers/DownloadTools.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
```

```
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.helpers;

import java.awt.image.BufferedImage;
import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.imageio.ImageIO;

/**
 * Provides static file-download functionalities
 *
 * @author Visa Nykänen
 */
public class DownloadTools {
    /**
     * Creates a png-file with the given filename from a byte array
     *
     * @param filename
     *            the filename for the image
     * @param img_bytes
     *            the byte-array containing the image
     * @return The image-file
     */
    public static File getImageFromByteArr(String filename, byte[] img_bytes) {
        ByteArrayInputStream bis = new ByteArrayInputStream(img_bytes);
        BufferedImage image;
        File outputfile = null;
        try {
            image = ImageIO.read(bis);
            bis.close();
            outputfile = File.createTempFile(filename, ".png");
            ImageIO.write(image, "png", outputfile);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return outputfile;
```

```
    }

    /**
     * Downloads the given file as the given MIME-type with the given name
     *
     * @param file
     *            the file to be downloaded
     * @param responseType
     *            the MIME-type of the file
     * @param downloadName
     *            The name with which the file is downloaded
     */
    public static void downloadFile(File file, String responseType, String
downloadName) {
        FacesContext fc = FacesContext.getCurrentInstance();
        ExternalContext ec = fc.getExternalContext();

        ec.responseReset();

        ec.setResponseContentType(responseType);
        ec.setResponseHeader("Content-Disposition", "attachment; filename=\"" +
downloadName + "\"");
        try {
            OutputStream outputStream = ec.getResponseOutputStream();
            FileInputStream input = new FileInputStream(file);
            byte[] buffer = new byte[1024];
            while ((input.read(buffer)) != -1) {
                outputStream.write(buffer);
            }
            outputStream.flush();
            input.close();
            fc.responseComplete();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     * Downloads the data given in a string as a UTF8-encoded csv-file
     *
     * @param data
     *            The data to be downloaded
     * @param fileName
     *            The name for the csv-file to be downloaded
     */
    public static void downloadCSV(String data, String fileName) {
        FacesContext fc = FacesContext.getCurrentInstance();
        ExternalContext ec = fc.getExternalContext();

        ec.responseReset();

        ec.setResponseContentType("text/csv");
        ec.setResponseHeader("Content-Disposition", "attachment; filename=\"" +
fileName + ".csv" + "\"");
        try {
            OutputStream outputStream = ec.getResponseOutputStream();
```

```
            byte[] bytes = data.getBytes("Windows-1252");
            for (byte b : bytes) {
                outputStream.write(b);
            }
            outputStream.flush();
            fc.responseComplete();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

# src/main/java/com/moveatis/interfaces/FeedbackAnalysis.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio, Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.interfaces;

import java.util.List;

import javax.ejb.Local;
```

```
import com.moveatis.feedbackanalysis.FeedbackAnalysisEntity;
import com.moveatis.records.FeedbackAnalysisRecordEntity;
import com.moveatis.user.AbstractUser;

/**
 * The interface for the feedback analysis enterprise java bean
 *
 * @author Visa Nykänen
 */
@Local(FeedbackAnalysis.class)
public interface FeedbackAnalysis {
    void create(FeedbackAnalysisEntity feedbackAnalysisEntity);

    void edit(FeedbackAnalysisEntity feedbackAnalysisEntity);

    void remove(FeedbackAnalysisEntity feedbackAnalysisEntity);

    void removeUnsavedObservation(FeedbackAnalysisEntity
feedbackAnalysisEntity);

    FeedbackAnalysisEntity find(Object id);

    List<FeedbackAnalysisEntity> findAll();

    List<FeedbackAnalysisEntity> findAllByObserver(AbstractUser analyzer);

    List<FeedbackAnalysisEntity> findWithoutEvent(AbstractUser analyzer);

    List<FeedbackAnalysisEntity> findByEventsNotOwned(AbstractUser analyzer);

    List<FeedbackAnalysisEntity> findRange(int[] range);

    List<FeedbackAnalysisRecordEntity> findRecords(Object id);

    int count();

    /**
     * Removes the given feedbackanalysisrecord from the given analysis and saves
     * the changes to the database Assumes that all the required changes such as
     * updating the ordernumbers of the remaining records have been done before
this
     * method is called
     *
     * @param feedbackAnalysis
     *            the analysis from which the record is removed
     * @param record
     *            the record to be removed
     */
    void removeRecordFromAnalysis(FeedbackAnalysisEntity feedbackAnalysis,
FeedbackAnalysisRecordEntity record);
}
```

# src/main/java/com/moveatis/interfaces/FeedbackAnalysisRecord.java

``````
```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.interfaces;

import java.util.List;

import com.moveatis.records.FeedbackAnalysisRecordEntity;

/**
 * the interface for the feedbackanalysisrecord enterprise java bean
 *
 * @author Visa Nykänen
 *
 */
public interface FeedbackAnalysisRecord {
    void create(FeedbackAnalysisRecordEntity feedbackAnalysisRecordEntity);

    void edit(FeedbackAnalysisRecordEntity feedbackAnalysisRecordEntity);

    void remove(FeedbackAnalysisRecordEntity feedbackAnalysisRecordEntity);

    FeedbackAnalysisRecordEntity find(Object id);
```
```

```
    List<FeedbackAnalysisRecordEntity> findAll();

    List<FeedbackAnalysisRecordEntity> findRange(int[] range);

    int count();
}
```

```
# src/main/java/com/moveatis/managedbeans/CategoryManagedBean.java
```

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.managedbeans;

import java.io.Serializable;
import java.util.Map;
import java.util.TreeMap;

import javax.faces.event.ActionEvent;
import javax.faces.view.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.moveatis.abstracts.AbstractCategoryEntity;
import com.moveatis.category.CategoryEntity;
import com.moveatis.category.CategorySetEntity;
import com.moveatis.interfaces.Category;
import com.moveatis.interfaces.Label;
import com.moveatis.label.LabelEntity;

/**
 * The bean that serves category management in the appropriate views.
 *
 * @author Sami Kallio <phinaliumz at outlook.com>
 */
@Named(value = "categoryManagedBean")
@ViewScoped
public class CategoryManagedBean implements Serializable {

    private static final Logger LOGGER =
LoggerFactory.getLogger(CategoryManagedBean.class);

    private static final long serialVersionUID = 1L;

    @Inject
    private Category categoryEJB;
    @Inject
    private Label labelEJB;
    @Inject
    private ControlManagedBean controlManagedBean;

    private String label;
    private String description;
    private Boolean canOverlap = false;

    /**
     * Creates a new instance of CategoryManagedBean.
     */
    public CategoryManagedBean() {

    }

    public String getLabel() {
        return label;
    }

    public void setLabel(String label) {
        this.label = label;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
```

```
    }

    public Boolean getCanOverlap() {
        return canOverlap;
    }

    public void setCanOverlap(Boolean canOverlap) {
        this.canOverlap = canOverlap;
    }

    public void addCategory(ActionEvent event) {
        LOGGER.debug("Category added");
    }

    /**
     * Creates a new category entity and adds it to the given category set.
     */
    public void createNewCategory(CategorySetEntity categorySetEntity) {
        CategoryEntity categoryEntity = new CategoryEntity();

        LabelEntity labelEntity = labelEJB.findByLabel(label);

        if (labelEntity == null) {
            labelEntity = new LabelEntity();
            labelEntity.setText(label);
            labelEJB.create(labelEntity);
        }

        categoryEntity.setLabel(labelEntity);
        categoryEntity.setCategorySet(categorySetEntity);
        categoryEntity.setCanOverlap(canOverlap);
        categoryEntity.setDescription(description);

        Map<Integer, AbstractCategoryEntity> categories =
categorySetEntity.getCategoryEntitys();

        if (categories == null) {
            categories = new TreeMap<>();
        }

        Integer orderNumber = categories.size();

        categories.put(orderNumber, categoryEntity);
        categoryEntity.setOrderNumber(orderNumber);
        categorySetEntity.setCategoryEntitys(categories);

        categoryEJB.create(categoryEntity);

        // controlManagedBean.addCategory(categoryEntity);
    }
}

``````

#
src/main/java/com/moveatis/managedbeans/FeebackAnalysisRecordTableManagedBean

.java

``````
```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.managedbeans;

import java.io.File;
import java.io.Serializable;
import java.text.MessageFormat;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.ResourceBundle;

import javax.annotation.PostConstruct;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.faces.view.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;

import org.primefaces.context.RequestContext;
```
```

```
import com.moveatis.abstracts.AbstractCategoryEntity;
import com.moveatis.feedbackanalysis.FeedbackAnalysisEntity;
import com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategoryEntity;
import
com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategorySetEntity;
import com.moveatis.helpers.DownloadTools;
import com.moveatis.interfaces.FeedbackAnalysis;
import com.moveatis.interfaces.MessageBundle;
import com.moveatis.records.FeedbackAnalysisRecordEntity;

/**
 * The managed bean in control of the record table for feedback analysis
 *
 * @author Tuomas Moisio
 */
@Named(value = "analysisRecordTable")
@ViewScoped
public class FeebackAnalysisRecordTableManagedBean implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Inject
    private FeedbackAnalysisManagedBean feedbackAnalysisManagedBean;

    private List<String> selectedSaveOptions;

    private FeedbackAnalysisEntity feedbackAnalysis;

    private String fileName;

    @Inject
    private FeedbackAnalysis feedbackAnalysisEJB;

    @Inject
    @MessageBundle
    private transient ResourceBundle messages;

    /**
     * The post constructor creates the feedback analysis
     */
    @PostConstruct
    protected void initialize() {
        feedbackAnalysisManagedBean.setIsTimerStopped(true);
        feedbackAnalysis =
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity();
    }

    /**
     * Gets the name of selected category from current category set
     *
     * @param selectedCategories
     *            Users selected categories
     * @param categorySet
```

```
     *            Category set in use
     * @return name of the selected category, empty if no category is selected
     */
    public String
getSelectedCategorysName(List<FeedbackAnalysisCategoryEntity>
selectedCategories,
            FeedbackAnalysisCategorySetEntity categorySet) {
        for (FeedbackAnalysisCategoryEntity cat_comp : selectedCategories) {
            for (AbstractCategoryEntity cat :
categorySet.getCategoryEntitys().values()) {
                if
(cat.getLabel().getText().contentEquals(cat_comp.getLabel().getText())
                        &&
cat.getCategorySet().getLabel().contentEquals(cat_comp.getCategorySet().getLa
bel())) {
                    return cat.getLabel().getText();
                }
            }
        }

        return "------";
    }

    /**
     * Gets the selected category from current category set
     *
     * @param selectedCategories
     *            Users selected categories
     * @param categorySet
     *            Category set in use
     * @return the selected category, new category if the category is not selected
     */
    public FeedbackAnalysisCategoryEntity
getSelectedCategory(List<FeedbackAnalysisCategoryEntity> selectedCategories,
            FeedbackAnalysisCategorySetEntity categorySet) {
        for (FeedbackAnalysisCategoryEntity cat_comp : selectedCategories) {
            for (AbstractCategoryEntity cat :
categorySet.getCategoryEntitys().values()) {
                if
(cat.getLabel().getText().contentEquals(cat_comp.getLabel().getText())
                        &&
cat.getCategorySet().getLabel().contentEquals(cat_comp.getCategorySet().getLa
bel())) {
                    return cat_comp;
                }
            }
        }
        FeedbackAnalysisCategoryEntity value = new
FeedbackAnalysisCategoryEntity();
        return value;
    }

    /**
     * Sends the user to the analyzer page with the selected record as main record.
     *
     * @param orderNumber
```

```
     *            order number of the selected record
     * @return String that faces-config uses to control navigation
     */
    public String edit(Integer orderNumber) {
        List<FeedbackAnalysisRecordEntity> list =
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity().getRecords();
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).getOrderNumber().intValue() ==
orderNumber.intValue()) {
                feedbackAnalysisManagedBean.setCurrentRecord(i + 1);
            }
        }
        return "editrow";
    }

    public List<String> getSelectedSaveOptions() {
        return selectedSaveOptions;
    }

    public void setSelectedSaveOptions(List<String> selectedSaveOptions) {
        this.selectedSaveOptions = selectedSaveOptions;
    }

    /**
     * Downloads the report-page table as an image.
     */
    public void downloadImage() {
        String fileName =
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity().getAnalysisName() +
"_report_";
        fileName = convertToFilename(fileName);

        File img = DownloadTools.getImageFromByteArr(fileName,
feedbackAnalysisManagedBean.getReportImage());
        DownloadTools.downloadFile(img, "image/png",
                img.getName().substring(0, img.getName().lastIndexOf("_")) +
".png");
        img.delete();
    }

    private static String convertToFilename(String s) {
        if (s == null || s.isEmpty()) {
            return "unnamed";
        }
        return s.replaceAll("[^a-zA-Z0-9_]", "_");
    }

    /**
     * Gets the confirmation message for deletion, includes the orderNumber of
the row to be deleted
     *
     * @param orderNumber
     * @return
     */
    public String getConfirm(int orderNumber) {
        FacesContext currentInstance = FacesContext.getCurrentInstance();
```

```
        return
MessageFormat.format(currentInstance.getApplication().getResourceBundle(curre
ntInstance, "msg").getString("repo_confirm"),orderNumber);
    }

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

}
```

#
src/main/java/com/moveatis/managedbeans/FeedbackAnalysisCategorySelectionMana
gedBean.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
```

```
package com.moveatis.managedbeans;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.ResourceBundle;
import java.util.Set;
import java.util.TreeMap;

import javax.annotation.PostConstruct;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.faces.view.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.moveatis.abstracts.AbstractCategoryEntity;
import com.moveatis.event.EventEntity;
import com.moveatis.event.EventGroupEntity;
import com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategoryEntity;
import
com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategorySetEntity;
import com.moveatis.helpers.Validation;
import com.moveatis.interfaces.CategorySet;
import com.moveatis.interfaces.EventGroup;
import com.moveatis.interfaces.MessageBundle;
import com.moveatis.interfaces.Session;
import com.moveatis.label.LabelEntity;
import com.moveatis.user.IdentifiedUserEntity;

/**
 * The managed bean in control of the category selection for feedback analysis
 *
 * @author Visa Nykänen
 */
@Named(value = "feedbackAnalysisCategorySelectionManagedBean")
@ViewScoped
public class FeedbackAnalysisCategorySelectionManagedBean implements
Serializable {
    private static final Logger LOGGER =
LoggerFactory.getLogger(FeedbackAnalysisCategorySelectionManagedBean.class);

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private String newFeedbackAnalysisCategorySetName;

    @Inject
```

```
    @MessageBundle // created MessageBundle to allow resourcebundle injection to
CDI beans
    private transient ResourceBundle messages; // RequestBundle is not
serializable

    private long selectedDefaultFeedbackAnalysisCategorySet;

    private long selectedPrivateFeedbackAnalysisCategorySet;

    private List<FeedbackAnalysisCategorySetEntity>
defaultFeedbackAnalysisCategorySets; // From group key or event that

                // was selected in control

                // page // page.
    private List<FeedbackAnalysisCategorySetEntity>
privateFeedbackAnalysisCategorySets;

    private List<FeedbackAnalysisCategorySetEntity>
feedbackAnalysisCategorySetsInUse;

    private EventGroupEntity eventGroup;

    @Inject
    private Session sessionEJB;

    @Inject
    private EventGroup eventGroupEJB;

    @Inject
    private FeedbackAnalysisManagedBean feedbackAnalysisManagedBean;

    /**
     * Gets the categorysets from the given groupkey, and if the user is logged
in
     * their own categorysets
     */
    @PostConstruct
    public void init() {
        eventGroup = null;
        defaultFeedbackAnalysisCategorySets = new
ArrayList<FeedbackAnalysisCategorySetEntity>();
        privateFeedbackAnalysisCategorySets = new
ArrayList<FeedbackAnalysisCategorySetEntity>();
        feedbackAnalysisCategorySetsInUse = new
ArrayList<FeedbackAnalysisCategorySetEntity>();
        if (feedbackAnalysisManagedBean.getEventEntity() != null) {
            EventEntity event = feedbackAnalysisManagedBean.getEventEntity();
            eventGroup = event.getEventGroup();
            if (eventGroup.getFeedbackAnalysisCategorySets() == null)
                eventGroup.setFeedbackAnalysisCategorySets(new
HashSet<FeedbackAnalysisCategorySetEntity>());

    defaultFeedbackAnalysisCategorySets.addAll(eventGroup.getFeedbackAnalysis
CategorySets());
        }
```

```
        if (sessionEJB.isIdentifiedUser()) {
            IdentifiedUserEntity user = sessionEJB.getLoggedIdentifiedUser();
            for (EventGroupEntity eg : eventGroupEJB.findAllForOwner(user))
                for (FeedbackAnalysisCategorySetEntity fba :
eg.getFeedbackAnalysisCategorySets())
                    privateFeedbackAnalysisCategorySets.add(fba);
        }

        List<FeedbackAnalysisCategorySetEntity> categorySets =
sessionEJB.getFeedbackAnalysisCategorySetsInUse();
        if (categorySets != null) {
            feedbackAnalysisCategorySetsInUse = categorySets;
        } else {
            for (FeedbackAnalysisCategorySetEntity categorySet :
defaultFeedbackAnalysisCategorySets) {
                feedbackAnalysisCategorySetsInUse.add(categorySet);
            }
        }
    }

    /**
     * Adds a new category set for the analysis if
     * newFeedbackAnalysisCategorySetName isn't empty.
     */
    public void addNewCategorySet() {
        String name =
Validation.validateForJsAndHtml(newFeedbackAnalysisCategorySetName);

        if (!name.isEmpty()) {
            for (FeedbackAnalysisCategorySetEntity set :
feedbackAnalysisCategorySetsInUse) {
                if (name.equals(set.getLabel())) {

    showErrorMessage(messages.getString("cs_errorNotUniqueCategorySet"));
                    return;
                }
            }

            FeedbackAnalysisCategorySetEntity categorySet = new
FeedbackAnalysisCategorySetEntity();
            categorySet.setLabel(name);
            Map<Integer, AbstractCategoryEntity> newCategoryEntities = new
TreeMap<Integer, AbstractCategoryEntity>();
            categorySet.setCategoryEntitys(newCategoryEntities);
            feedbackAnalysisCategorySetsInUse.add(categorySet);
            newFeedbackAnalysisCategorySetName = "";
        }
    }

    /**
     * Adds a new category to the given categoryset
     *
     * @param categorySet
     *            the categoryset to which the new category is added
     */
```

```
    public void addNewCategoryToCategorySet(FeedbackAnalysisCategorySetEntity
categorySet) {
        FeedbackAnalysisCategoryEntity fac = new
FeedbackAnalysisCategoryEntity();
        fac.setLabel(new LabelEntity());

        Map<Integer, AbstractCategoryEntity> categories =
categorySet.getCategoryEntitys();
        fac.setOrderNumber(categories.keySet().size());
        fac.setCategorySet(categorySet);
        categories.put(categories.keySet().size(), fac);
    }

    /**
     * removes the given category from the given categoryset
     *
     * @param categorySet
     *            the categoryset from which a category is to be removed
     * @param category
     *            the category to be removed
     */
    public void
removeCategoryFromCategorySet(FeedbackAnalysisCategorySetEntity categorySet,
            FeedbackAnalysisCategoryEntity category) {

        Map<Integer, AbstractCategoryEntity> categories =
categorySet.getCategoryEntitys();
        Map<Integer, AbstractCategoryEntity> tmp_categories = new
TreeMap<Integer, AbstractCategoryEntity>();

        categories.remove(category.getOrderNumber());
        List<Integer> keys=new ArrayList<>();
        keys.addAll(categories.keySet());
        Collections.sort(keys);
        int i = 0;
        for (int key : keys) {
            categories.get(key).setOrderNumber(i);
            tmp_categories.put(i, categories.get(key));
            i++;
        }
        categorySet.setCategoryEntitys(tmp_categories);
    }

    /**
     * Finds the categoryset with the given id from the given list
     *
     * @param categorySets
     *            list of categorysets
     * @param id
     *            the ID of the categoryset that needs to be accessed
     * @return the found categoryset or a new categoryset if one with the given
ID
     *         isn't found
     */
    private FeedbackAnalysisCategorySetEntity
findById(List<FeedbackAnalysisCategorySetEntity> categorySets, long id) {
```

```
        for (FeedbackAnalysisCategorySetEntity facs : categorySets) {
            if (facs.getId() == id) {
                return (facs);
            }
        }
        return new FeedbackAnalysisCategorySetEntity();
    }

    /**
     * Adds the selected default category set for the analysis.
     */
    public void addDefaultCategorySet() {
        FeedbackAnalysisCategorySetEntity sdc =
findById(defaultFeedbackAnalysisCategorySets,
                selectedDefaultFeedbackAnalysisCategorySet);
        if (!feedbackAnalysisCategorySetsInUse.contains(sdc)) {
            feedbackAnalysisCategorySetsInUse.add(sdc);
        }
    }

    /**
     * Adds the selected private category set for the analysis.
     */
    public void addPrivateCategorySet() {
        FeedbackAnalysisCategorySetEntity spc =
findById(privateFeedbackAnalysisCategorySets,
                selectedPrivateFeedbackAnalysisCategorySet);
        if (!feedbackAnalysisCategorySetsInUse.contains(spc)) {
            feedbackAnalysisCategorySetsInUse.add(spc);
        }
    }

    /**
     * Removes the given categoryset from use
     *
     * @param categorySet
     *            the categoryset to be removed
     */
    public void removeCategorySet(int categorySet) {
        feedbackAnalysisCategorySetsInUse.remove(categorySet);
    }

    /**
     * Checks if the continue button should be disabled. The button is disabled
if
     * no category sets have been added for the observation or if some of the added
     * category sets are empty.
     *
     * @return True if the continue button should be disabled.
     */
    public boolean isContinueDisabled() {
        for (FeedbackAnalysisCategorySetEntity categorySet :
feedbackAnalysisCategorySetsInUse) {
            if (categorySet.getCategoryEntitys().isEmpty())
                return true;
        }
```

```
        return feedbackAnalysisCategorySetsInUse.isEmpty();
    }

    /**
     * Shows given error message in primefaces message popup.
     *
     * @param message
     *            Error message to show.
     */
    private void showErrorMessage(String message) {
        FacesContext context = FacesContext.getCurrentInstance();
        context.addMessage(null,
                new FacesMessage(FacesMessage.SEVERITY_ERROR,
messages.getString("dialogErrorTitle"), message));
    }

    /**
     * Checks the categories in use before letting the user continue to
observation.
     * The categories in the same category set should have different names. The
     * categories shouldn't have empty names. At least one category should be
     * selected for the observation. It shows an error message if the categories
     * aren't ok.
     *
     * @return "analysiscategoriesok" if the categories were ok, otherwise an
empty
     *         string.
     */
    public String checkCategories() {
        boolean atLeastOneCategorySelected = false;

        for (FeedbackAnalysisCategorySetEntity categorySet :
feedbackAnalysisCategorySetsInUse) {

            Map<Integer, AbstractCategoryEntity> categories =
categorySet.getCategoryEntitys();

            if (hasDuplicate(categories)) {

    showErrorMessage(messages.getString("cs_errorNotUniqueCategories"));
                return "";
            }

            if (!categories.isEmpty()) {
                atLeastOneCategorySelected = true;
            } else {

    showErrorMessage(messages.getString("cs_warningEmptyCategorySets"));
                return "";
            }

            for (AbstractCategoryEntity category : categories.values()) {

                if (category.getLabel().getText().isEmpty()) {

    showErrorMessage(messages.getString("cs_warningEmptyCategories"));
```

```
                    return "";
                }
            }
        }

        if (!atLeastOneCategorySelected) {
            showErrorMessage(messages.getString("cs_errorNoneSelected"));
            return "";
        }

    feedbackAnalysisManagedBean.setFeedbackAnalysisCategorySetsInUse(feedback
AnalysisCategorySetsInUse);
        feedbackAnalysisManagedBean.setFeedbackAnalysisEntity(null);
        feedbackAnalysisManagedBean.init();
        feedbackAnalysisManagedBean.setIsTimerEnabled(isTimerEnabled);

        return "analysiscategoriesok";
    }

    /**
     * Checks if given categories contain duplicate names.
     *
     * @param categories
     *            List of categories to check.
     * @return True if categories contain duplicates, otherwise false.
     */
    private static boolean hasDuplicate(Map<Integer, AbstractCategoryEntity>
categories) {
        Set<String> set = new HashSet<>();
        for (AbstractCategoryEntity category : categories.values()) {
            String name = category.getLabel().getText();
            if (!name.isEmpty() && !set.add(name)) {
                return true;
            }
        }
        return false;
    }

    private boolean isTimerEnabled;

    public boolean getIsTimerEnabled() {
        return isTimerEnabled;
    }

    public void setIsTimerEnabled(boolean timerEnabled) {
        isTimerEnabled = timerEnabled;
    }

    public String getNewFeedbackAnalysisCategorySetName() {
        return newFeedbackAnalysisCategorySetName;
    }

    public void setNewFeedbackAnalysisCategorySetName(String
newFeedbackAnalysisCategorySetName) {
        this.newFeedbackAnalysisCategorySetName =
newFeedbackAnalysisCategorySetName;
```

```
    }

    public long getSelectedDefaultFeedbackAnalysisCategorySet() {
        return selectedDefaultFeedbackAnalysisCategorySet;
    }

    public void setSelectedDefaultFeedbackAnalysisCategorySet(long
selectedDefaultFeedbackAnalysisCategorySet) {
        this.selectedDefaultFeedbackAnalysisCategorySet =
selectedDefaultFeedbackAnalysisCategorySet;
    }

    public long getSelectedPrivateFeedbackAnalysisCategorySet() {
        return selectedPrivateFeedbackAnalysisCategorySet;
    }

    public void setSelectedPrivateFeedbackAnalysisCategorySet(long
selectedPrivateFeedbackAnalysisCategorySet) {
        this.selectedPrivateFeedbackAnalysisCategorySet =
selectedPrivateFeedbackAnalysisCategorySet;
    }

    public List<FeedbackAnalysisCategorySetEntity>
getDefaultFeedbackAnalysisCategorySets() {
        return defaultFeedbackAnalysisCategorySets;
    }

    public void setDefaultFeedbackAnalysisCategorySets(
            List<FeedbackAnalysisCategorySetEntity>
defaultFeedbackAnalysisCategorySets) {
        this.defaultFeedbackAnalysisCategorySets =
defaultFeedbackAnalysisCategorySets;
    }

    public List<FeedbackAnalysisCategorySetEntity>
getPrivateFeedbackAnalysisCategorySets() {
        return privateFeedbackAnalysisCategorySets;
    }

    public void setPrivateFeedbackAnalysisCategorySets(
            List<FeedbackAnalysisCategorySetEntity>
privateFeedbackAnalysisCategorySets) {
        this.privateFeedbackAnalysisCategorySets =
privateFeedbackAnalysisCategorySets;
    }

    public List<FeedbackAnalysisCategorySetEntity>
getFeedbackAnalysisCategorySetsInUse() {
        return feedbackAnalysisCategorySetsInUse;
    }

    public void setFeedbackAnalysisCategorySetsInUse(
            List<FeedbackAnalysisCategorySetEntity>
feedbackAnalysisCategorySetsInUse) {
        this.feedbackAnalysisCategorySetsInUse =
feedbackAnalysisCategorySetsInUse;
```

```
    }

    public EventGroupEntity getEventGroup() {
        return eventGroup;
    }

    public void setEventGroup(EventGroupEntity eventGroup) {
        this.eventGroup = eventGroup;
    }

    public FeedbackAnalysisManagedBean getFeedbackAnalysisManagedBean() {
        return feedbackAnalysisManagedBean;
    }

    public void setFeedbackAnalysisManagedBean(FeedbackAnalysisManagedBean
feedbackAnalysisManagedBean) {
        this.feedbackAnalysisManagedBean = feedbackAnalysisManagedBean;
    }

}
```

# src/main/java/com/moveatis/managedbeans/FeedbackAnalysisManagedBean.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
```

```
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.managedbeans;

import java.io.Serializable;
import java.text.DateFormat;
import java.text.MessageFormat;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.ResourceBundle;

import javax.ejb.EJB;
import javax.enterprise.context.SessionScoped;
import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.ValidatorException;
import javax.inject.Inject;
import javax.inject.Named;

import org.jboss.logging.annotations.Message;
import org.primefaces.context.RequestContext;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.moveatis.abstracts.AbstractCategoryEntity;
import com.moveatis.event.EventEntity;
import com.moveatis.feedbackanalysis.FeedbackAnalysisEntity;
import com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategoryEntity;
import
com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategorySetEntity;
import com.moveatis.interfaces.CategorySet;
import com.moveatis.interfaces.FeedbackAnalysisRecord;
import com.moveatis.interfaces.FeedbackAnalysis;
import com.moveatis.interfaces.Label;
import com.moveatis.interfaces.Session;
import com.moveatis.label.LabelEntity;
import com.moveatis.records.FeedbackAnalysisRecordEntity;

/**
 * The managed bean controlling the feedbackanalysis in view TODO: extract the
 * methods concerning only a certain view to new managed beans controlling said
 * views (mostly analyzer, but some recordtable and summary functionalities too)
 *
 * @author Visa Nykänen
 *
 */
@Named(value = "feedbackAnalysisManagedBean")
@SessionScoped
public class FeedbackAnalysisManagedBean implements Serializable {

    private static final Logger LOGGER =
```

```
LoggerFactory.getLogger(ObservationManagedBean.class);

    private static final long serialVersionUID = 1L;

    @Inject
    private Label labelEJB;

    /**
     * The feedbackanalysisentity being edited
     */
    private FeedbackAnalysisEntity feedbackAnalysisEntity;

    /**
     * The categorysets being used in the analysis event
     */
    private List<FeedbackAnalysisCategorySetEntity>
feedbackAnalysisCategorySetsInUse;

    /**
     * The index(+1) of the record currently in view
     */

    private int currentRecordNumber;

    /**
     * If the analysis has some new categorysets they need to be saved, so
     * CategorySetBean is needed
     */
    @Inject
    private CategorySet categorySetEJB;

    /**
     * The record currently in view
     */
    private FeedbackAnalysisRecordEntity currentRecord;

    /**
     * The event the analysis is performed for
     */
    private EventEntity eventEntity;

    /**
     * used to save the analysis to the database
     */
    @EJB
    private FeedbackAnalysis feedbackAnalysisEJB;

    /**
     * The categorysets are gotten from the session
     */
    @Inject
    private Session sessionBean;

    private FeedbackAnalysisCategoryEntity selectedCategory;

    /**
```

```
     * If new records are added to the analysis after it has already been saved
to
     * the database the records need to be saved individually so
     * feedbackanalysisrecordbean is needed
     */
    @Inject
    private FeedbackAnalysisRecord feedbackAnalysisRecordEJB;

    /**
     * The timer value, set to be the duration of the analysis once navigating
to
     * the record table
     */
    private long duration;

    /**
     * Whether the timer is stopped
     */
    private boolean isTimerStopped;

    private boolean isTimerEnabled;

    @Inject
    private UserManagedBean userManagedBean;

    private byte[] pieImage, tableImage, barImage, reportImage;

    private String reportCSV;

    public void setIsTimerEnabled(boolean timerEnabled) {
        this.isTimerEnabled = timerEnabled;
    }

    public boolean getIsTimerEnabled() {
        return isTimerEnabled;
    }

    public void setBarImage(byte[] img) {
        barImage = img;
    }

    public byte[] getBarImage() {
        return barImage;
    }

    public void setTableImage(byte[] img) {
        tableImage = img;
    }

    public byte[] getTableImage() {
        return tableImage;
    }

    public void setPieImage(byte[] img) {
        pieImage = img;
    }
```

```
    public byte[] getPieImage() {
        return pieImage;
    }

    public void setReportCSV(String data) {
        reportCSV = data;
    }

    public String getReportCSV() {
        return reportCSV;
    }

    public void setReportImage(byte[] img) {
        this.reportImage = img;
    }

    public byte[] getReportImage() {
        return reportImage;
    }

    public void setEventEntity(EventEntity eventEntity) {
        this.eventEntity = eventEntity;
    }

    public EventEntity getEventEntity() {
        return this.eventEntity;
    }

    public FeedbackAnalysisEntity getFeedbackAnalysisEntity() {
        return feedbackAnalysisEntity;
    }

    public void setFeedbackAnalysisEntity(FeedbackAnalysisEntity
feedbackAnalysisEntity) {
        this.feedbackAnalysisEntity = feedbackAnalysisEntity;
    }

    public void setFeedbackAnalysisName(String name) {
        this.feedbackAnalysisEntity.setAnalysisName(name);
    }

    public void setFeedbackAnalysisDuration(long duration) {
        this.feedbackAnalysisEntity.setDuration(duration);
    }

    public int getCurrentRecordNumber() {
        return currentRecordNumber;
    }

    public void setCurrentRecordNumber(int currentRecordNumber) {
        this.currentRecordNumber = currentRecordNumber;
    }

    public boolean getIsTimerStopped() {
        return isTimerStopped;
```

```
    }

    public void setIsTimerStopped(boolean timerStopped) {
        this.isTimerStopped = timerStopped;
    }

    public void pauseContinue() {
        isTimerStopped = !isTimerStopped;
    }

    public String getDurationAsString() {
        return getLongAsTimeStamp(duration);
    }

    public void setDuration(long duration) {
        this.duration = duration;
    }

    public void setSelectedCategory(FeedbackAnalysisCategoryEntity
selectedCategory) {
        this.selectedCategory = selectedCategory;
    }

    public FeedbackAnalysisCategoryEntity getSelectedCategory() {
        return selectedCategory;
    }

    public FeedbackAnalysisRecordEntity getCurrentRecord() {
        return currentRecord;
    }

    public List<FeedbackAnalysisCategorySetEntity>
getFeedbackAnalysisCategorySetsInUse() {
        return feedbackAnalysisCategorySetsInUse;
    }

    public void setFeedbackAnalysisCategorySetsInUse(
            List<FeedbackAnalysisCategorySetEntity>
feedbackAnalysisCategorySetsInUse) {
        this.feedbackAnalysisCategorySetsInUse =
feedbackAnalysisCategorySetsInUse;
    }

    /**
     * Initializes all the necessary information for the analysis
     */
    public void init() {
        if (feedbackAnalysisEntity == null) {
            Locale locale = userManagedBean.getLocale();
            ResourceBundle messages =
ResourceBundle.getBundle("com.moveatis.messages.Messages", locale);
            currentRecordNumber = 1;
            feedbackAnalysisEntity = new FeedbackAnalysisEntity();
            feedbackAnalysisEntity.setCreated();

            DateFormat dateFormat =
```

```
DateFormat.getDateTimeInstance(DateFormat.SHORT, DateFormat.SHORT, locale);
            dateFormat.setTimeZone(sessionBean.getSessionTimeZone());


    feedbackAnalysisEntity.setAnalysisName(messages.getString("asum_anatitle"
) + " - "
                    + dateFormat.format(feedbackAnalysisEntity.getCreated()));
            feedbackAnalysisEntity.setRecords(new
ArrayList<FeedbackAnalysisRecordEntity>());
            currentRecord = new FeedbackAnalysisRecordEntity();
            setOrderNumberForRecord();
            if (feedbackAnalysisCategorySetsInUse != null)
                for (FeedbackAnalysisCategorySetEntity facs :
feedbackAnalysisCategorySetsInUse)
                    for (AbstractCategoryEntity fac :
facs.getCategoryEntitys().values())
                        ((FeedbackAnalysisCategoryEntity)
fac).setInRecord(false);
            currentRecord.setFeedbackAnalysis(feedbackAnalysisEntity);
            feedbackAnalysisEntity.addRecord(currentRecord);
        } else
            setCurrentRecord(feedbackAnalysisEntity.getRecords().size());
        isTimerStopped = true;
        duration = feedbackAnalysisEntity.getDuration();
    }

    /**
     * Resets the start time selected categories and the comment for the record
     * currently in view
     */
    public void resetCurrentRecord() {
        currentRecord.setComment(null);
        currentRecord.setStartTime(null);
        resetSelectedCategories();
        editRecord();
    }

    /**
     * Finds the next record after the currently viewed one with timestamp set
and
     * returns its timestamp. If no such record exists returns currently elapsed
     * time.
     *
     * @return The maximum value for the currently shown records timestamp
     */
    public long getMaxTimeStampForCurrentRecord() {
        if (currentRecordNumber == feedbackAnalysisEntity.getRecords().size())
            return duration;
        for (int i = currentRecordNumber + 1; i <=
feedbackAnalysisEntity.getRecords().size(); i++) {
            Long start = findRecordByOrderNumber(i).getStartTime();
            if (start != null && start > 0)
                return start;
        }
        return duration;
    }
```

```
    /**
     * Finds the previous record before the currently viewed one with timestamp
set
     * and returns its timestamp. If no such record exists returns 0.
     *
     * @return The minimum value for the currently shown records timestamp
     */
    public long getMinTimeStampForCurrentRecord() {
        if (currentRecordNumber == 1)
            return 0;
        for (int i = currentRecordNumber - 1; i >= 1; i--) {
            Long start = findRecordByOrderNumber(i).getStartTime();
            if (start != null && start > 0)
                return start;
        }
        return 0;
    }

    /**
     * Returns the given number of seconds in a string showing the minutes and
     * seconds
     *
     * @param seconds
     *            the value as seconds
     * @return the timestamp as the amount of minutes and seconds in a string
     */
    public String getLongAsTimeStamp(long seconds) {
        if (seconds == 0)
            return "--:--";
        return String.format("%02d:%02d", (seconds / 60), (seconds % 60));

    }

    /**
     * increments the timer value every second if the timer is running
     */
    public void increment() {
        if (!isTimerStopped)
            duration += 1;
    }

    /**
     * Sets the currently shown record to be the record given in the parameter
     *
     * @param currentRecord
     *            the record to be shown
     */
    public void setCurrentRecord(FeedbackAnalysisRecordEntity currentRecord) {
        currentRecordNumber = currentRecord.getOrderNumber();
        for (FeedbackAnalysisCategorySetEntity facs :
feedbackAnalysisCategorySetsInUse)
            for (AbstractCategoryEntity fac :
facs.getCategoryEntitys().values())
                ((FeedbackAnalysisCategoryEntity) fac).setInRecord(false);
```

```
        List<FeedbackAnalysisCategoryEntity> selectedCategories =
currentRecord.getSelectedCategories();
        for (FeedbackAnalysisCategoryEntity category : selectedCategories)
            category.setInRecord(true);
        this.currentRecord = currentRecord;
    }

    /**
     * Sets the starttime of the currently viewed record based on the timer value
if
     * the record isn't in between other records and its starttime hasn't already
     * been set
     */
    public void setTimeStamp() {
        if (currentRecord.getStartTime() == null && currentRecordNumber ==
feedbackAnalysisEntity.getRecords().size())
            currentRecord.setStartTime(duration);
        editRecord();
    }

    /**
     * Finds the record in the feedbackanalysis based on its ordernumber
     *
     * @param orderNumber
     *            the ordernumber of the record to be accessed
     * @return the record with the given ordernumber
     */
    private FeedbackAnalysisRecordEntity findRecordByOrderNumber(Integer
orderNumber) {
        List<FeedbackAnalysisRecordEntity> records =
feedbackAnalysisEntity.getRecords();
        for (FeedbackAnalysisRecordEntity record : records)
            if (record.getOrderNumber() == orderNumber)
                return record;
        return new FeedbackAnalysisRecordEntity();
    }

    /**
     * Sets the ordernumber for the currently edited record If the record is added
     * between records, sets the following records ordernumbers to be one higher
     */
    private void setOrderNumberForRecord() {
        for (int i = feedbackAnalysisEntity.getRecords().size(); i >=
currentRecordNumber; i--)
            findRecordByOrderNumber(i).setOrderNumber(i + 1);
        currentRecord.setOrderNumber(currentRecordNumber);
    }

    /**
     * Saves the changes made to the current record if it exists, then sets record
     * to be shown in the view based on the given ordernumber
     *
     * @param recordNumber
     *            The ordernumber of the record to be accessed
     */
    public void setCurrentRecord(int recordNumber) {
```

```
        if (currentRecord != null)
            editRecord();
        resetSelectedCategories();
        currentRecordNumber = recordNumber;
        currentRecord = findRecordByOrderNumber(recordNumber);
        List<FeedbackAnalysisCategoryEntity> selectedCategories =
currentRecord.getSelectedCategories();
        for (FeedbackAnalysisCategorySetEntity facs :
feedbackAnalysisCategorySetsInUse)
            for (AbstractCategoryEntity fac :
facs.getCategoryEntitys().values()) {
                ((FeedbackAnalysisCategoryEntity) fac).setInRecord(false);
                for (FeedbackAnalysisCategoryEntity category :
selectedCategories)
                    if
(category.getCategorySet().getLabel().contentEquals(fac.getCategorySet().getL
abel())
                            &&
category.getLabel().getText().contentEquals(fac.getLabel().getText())) {
                        ((FeedbackAnalysisCategoryEntity)
fac).setInRecord(true);
                        break;
                    }

            }

    }

    /**
     * Tells whether there are records before or after the one currently shown
based
     * on the parameter.
     *
     * @param isLeft
     *            whether to check before or after the current record
     * @return whether there are records before or after the one currently shown
     */
    public boolean isNavigationDisabled(boolean isLeft) {
        if (isLeft)
            return currentRecordNumber == 1;
        else
            return currentRecordNumber ==
feedbackAnalysisEntity.getRecords().size();
    }

    /**
     * saves the changes to the record currently in view and initializes a new
     * record
     */
    public void addRecord() {
        if (feedbackAnalysisEntity == null)
            feedbackAnalysisEntity = new FeedbackAnalysisEntity();

        editRecord();
        resetSelectedCategories();
```

```
        currentRecord = new FeedbackAnalysisRecordEntity();
        currentRecord.setSelectedCategories(new
ArrayList<FeedbackAnalysisCategoryEntity>());
        currentRecordNumber++;
        setOrderNumberForRecord();
        currentRecord.setFeedbackAnalysis(feedbackAnalysisEntity);
        feedbackAnalysisEntity.addRecord(currentRecord);
    }

    /**
     * Resets the selected categories to not be selected
     */
    private void resetSelectedCategories() {
        for (FeedbackAnalysisCategorySetEntity facs :
feedbackAnalysisCategorySetsInUse)
            for (AbstractCategoryEntity fac :
facs.getCategoryEntitys().values())
                if (((FeedbackAnalysisCategoryEntity) fac).getInRecord()) {
                    ((FeedbackAnalysisCategoryEntity) fac).setInRecord(false);
                }
    }

    /**
     * Saves the changes made to the categories in the currently shown record
     */
    private void editRecord() {
        List<FeedbackAnalysisCategoryEntity> selectedCategories = new
ArrayList<FeedbackAnalysisCategoryEntity>();
        for (FeedbackAnalysisCategorySetEntity facs :
feedbackAnalysisCategorySetsInUse)
            for (AbstractCategoryEntity fac :
facs.getCategoryEntitys().values())
                if (((FeedbackAnalysisCategoryEntity) fac).getInRecord()) {
                    selectedCategories.add((FeedbackAnalysisCategoryEntity)
fac);
                }
        currentRecord.setSelectedCategories(selectedCategories);
    }

    /**
     * Checks if the analysis has any categories selected, prevents the user from
     * making empty analyses.
     *
     * @return True if there are no categories selected in any of the records,
     *         otherwise false
     */
    public boolean checkNoCategoriesSelected() {
        for (FeedbackAnalysisRecordEntity record :
feedbackAnalysisEntity.getRecords()) {
            if (record.getSelectedCategories() == null ||
record.getSelectedCategories().size() == 0)
                continue;
            else
                return false;
        }
        return true;
```

```
    }

    /**
     * Checks if the analysis has records with no categories selected. The user
is
     * asked to confirm that they want to continue if that is the case
     *
     * @return True if there is at least one record that has not been classified
     *         based on at least one categoryset, false otherwise
     */
    public boolean containsUnclassifiedRecords() {
        for (FeedbackAnalysisRecordEntity record :
feedbackAnalysisEntity.getRecords()) {
            if (feedbackAnalysisCategorySetsInUse.size() >
record.getSelectedCategories().size())
                return true;
        }
        return false;
    }

    /**
     * navigates to the summary page
     *
     * @return the key string that is used by facesconfig.xml to navigate to the
     *         correct page
     */
    public String toSummary() {
        if (checkNoCategoriesSelected()) {
            Locale locale = userManagedBean.getLocale();
            ResourceBundle messages =
ResourceBundle.getBundle("com.moveatis.messages.Messages", locale);
            RequestContext.getCurrentInstance().showMessageInDialog(new
FacesMessage(FacesMessage.SEVERITY_ERROR,
                    messages.getString("ana_continuefailheader"),
messages.getString("ana_continuefail")));
            return "";
        }
        return "summary";
    }

    /**
     * Validates the current record, checks if it has no selected categories or
no
     * written feedback and throws an error if that is the case
     *
     * @param context
     * @param component
     * @param value
     */
    public void validateNonEmptyRecord(FacesContext context, UIComponent
component, Object value) {
        Locale locale = userManagedBean.getLocale();
        ResourceBundle messages =
ResourceBundle.getBundle("com.moveatis.messages.Messages", locale);
        String message = messages.getString("ana_emptyRecord");
        if (isRecordEmpty(currentRecord))
```

```
            throw new ValidatorException(
                    new FacesMessage(FacesMessage.SEVERITY_ERROR,
messages.getString("dialogErrorTitle"), message));
    }

    /**
     * Checks whether the given record has categories selected
     *
     * @param record
     *            the record to check
     * @return whether the given record has categories selected
     */
    private boolean isRecordEmpty(FeedbackAnalysisRecordEntity record) {
        return (record.getSelectedCategories() == null ||
record.getSelectedCategories().isEmpty())
                && (record.getComment() == null ||
record.getComment().isEmpty());
    }

    /**
     * Checks if there are empty records returns the order number of the first
found
     * empty record, if they exist -1 if all no empty records are present
     *
     * @return order number of the first empty record
     */
    private int getEmptyRecordNumber() {
        for (FeedbackAnalysisRecordEntity rec :
feedbackAnalysisEntity.getRecords())
            if (isRecordEmpty(rec))
                return rec.getOrderNumber();
        return -1;
    }

    /**
     * Makes sure changes to the currently shown record are saved, stops the timer,
     * sets the duration of the analysis and checks wheter there are some empty
     * records and whether none of the records has any categories selected
navigates
     * to the recordtable page if not.
     *
     * @return the key string that is used by facesconfig.xml to navigate to the
     *         correct page
     */
    public String toRecordTable() {
        Locale locale = userManagedBean.getLocale();
        ResourceBundle messages =
ResourceBundle.getBundle("com.moveatis.messages.Messages", locale);
        editRecord();
        if (checkNoCategoriesSelected()) {
            RequestContext.getCurrentInstance().showMessageInDialog(new
FacesMessage(FacesMessage.SEVERITY_ERROR,
                    messages.getString("ana_continuefailheader"),
messages.getString("ana_continuefail")));
            return "";
        }
```

```
        if (getEmptyRecordNumber() > 0) {
            RequestContext.getCurrentInstance().showMessageInDialog(new
FacesMessage(FacesMessage.SEVERITY_ERROR,
                    messages.getString("ana_continuefailheader"),

    MessageFormat.format(messages.getString("ana_emptyRecordWithNumber"),
getEmptyRecordNumber())));
            return "";
        }

        feedbackAnalysisEntity.setDuration(duration);
        isTimerStopped = true;
        return "recordtable";
    }

    /**
     * Delete's the selected record from the datatable and the database, if the
     * analysis has already been saved.
     *
     * @param record
     *            selected row
     */
    public void delete(int orderNumber) {
        Locale locale = userManagedBean.getLocale();
        ResourceBundle messages =
ResourceBundle.getBundle("com.moveatis.messages.Messages", locale);
        if (feedbackAnalysisEntity.getRecords().size() == 1) {
            RequestContext.getCurrentInstance().showMessageInDialog(new
FacesMessage(FacesMessage.SEVERITY_ERROR,
                    messages.getString("repo_deletefailheader"),
messages.getString("repo_deletefail")));
            return;
        }

        List<FeedbackAnalysisRecordEntity> list =
feedbackAnalysisEntity.getRecords();
        FeedbackAnalysisRecordEntity record = null;
        for (int i = 0; i < list.size(); i++) {
            if (list.get(i).getOrderNumber() != null &&
list.get(i).getOrderNumber().intValue() == orderNumber) {
                record = list.get(i);
                list.remove(i);
                break;
            }
        }
        setOrderNumbers(list);
        feedbackAnalysisEntity.setRecords(list);
        if (record != null && record.getId() != null)

    feedbackAnalysisEJB.removeRecordFromAnalysis(feedbackAnalysisEntity,
record);
        if (orderNumber > list.size())
            setCurrentRecord(list.size());
        else
            setCurrentRecord(orderNumber);
    }
```

```
    /**
     * Updates order numbers to records list.
     *
     * @param list
     *            users records
     */
    private void setOrderNumbers(List<FeedbackAnalysisRecordEntity> list) {
        list.sort((a, b) -> a.getOrderNumber().compareTo(b.getOrderNumber()));
        Integer newOrderNumber = 1;
        for (int i = 0; i < list.size(); i++) {
            list.get(i).setOrderNumber(i + 1);
            newOrderNumber++;
        }
    }

    /**
     * The method saves the analysis to the database. Copies are made of all the
     * categorysets used by the analysis, so that later edits to the categorysets
     * won't affect old analyses.
     */
    public void saveFeedbackAnalysis() {
        if (feedbackAnalysisEntity.getId() == null) {

            for (FeedbackAnalysisCategorySetEntity categorySet :
feedbackAnalysisCategorySetsInUse) {

                if (categorySet.getId() != null)
                    categorySetEJB.detachCategorySet(categorySet);

                for (AbstractCategoryEntity cat :
categorySet.getCategoryEntitys().values()) {
                    LabelEntity label =
labelEJB.findByLabel(cat.getLabel().getText());
                    cat.setCategorySet(categorySet);
                    if (label == null) {
                        cat.setLabel(new
LabelEntity(cat.getLabel().getText()));
                        labelEJB.create(cat.getLabel());
                    } else
                        cat.setLabel(label);
                }
                categorySetEJB.create(categorySet);
            }
            feedbackAnalysisEntity.setEvent(eventEntity);

    feedbackAnalysisEntity.setObserver(sessionBean.getLoggedIdentifiedUser())
;
            feedbackAnalysisEJB.create(feedbackAnalysisEntity);
        } else {
            for (FeedbackAnalysisRecordEntity record :
feedbackAnalysisEntity.getRecords()) {
                if (record.getId() == null)
                    feedbackAnalysisRecordEJB.create(record);
                else
                    feedbackAnalysisRecordEJB.edit(record);
```

```
            }
            feedbackAnalysisEJB.edit(feedbackAnalysisEntity);
        }
    }

    /**
     * Sets the categorysetsinuse to be null,
     */
    public void resetCategorySetsInUse() {
        this.feedbackAnalysisCategorySetsInUse = null;
    }

}
``````

#
src/main/java/com/moveatis/managedbeans/FeedbackAnalysisSummaryManagedBean.ja
va

``````
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.managedbeans;
```

```
import static org.primefaces.model.chart.LegendPlacement.OUTSIDE;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Serializable;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
import java.util.ResourceBundle;

import javax.annotation.PostConstruct;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.faces.view.ViewScoped;
import javax.inject.Inject;
import javax.inject.Named;

import org.primefaces.model.chart.Axis;
import org.primefaces.model.chart.AxisType;
import org.primefaces.model.chart.BarChartModel;
import org.primefaces.model.chart.ChartSeries;
import org.primefaces.model.chart.PieChartModel;

import com.moveatis.abstracts.AbstractCategoryEntity;
import com.moveatis.feedbackanalysis.FeedbackAnalysisEntity;
import com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategoryEntity;
import
com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategorySetEntity;
import com.moveatis.helpers.DownloadTools;
import com.moveatis.interfaces.Mailer;
import com.moveatis.interfaces.MessageBundle;
import com.moveatis.records.FeedbackAnalysisRecordEntity;

/**
 * The managed bean to control the summary page
 *
 * @author Visa Nykänen, Tuomas Moisio
 *
 */
@Named(value = "feedbackAnalysisSummaryManagedBean")
@ViewScoped
public class FeedbackAnalysisSummaryManagedBean implements Serializable {

    /**
     * A helper class only useful in this view to hold the information used by
the
     * summary table
     *
     * @author Visa Nykänen
     */
    public class TableInformation {
```

```
        private String feedbackAnalysisCategorySet;

        private List<String> categories;

        private List<Integer> counts;

        public TableInformation(String feedbackAnalysisCategorySet) {
            this.categories = new ArrayList<String>();
            this.counts = new ArrayList<Integer>();
            this.setFeedbackAnalysisCategorySet(feedbackAnalysisCategorySet);
        }

        private void addCategoryWithCount(String category, Integer count) {
            this.categories.add(category);
            this.counts.add(count);
        }

        public String getFeedbackAnalysisCategorySet() {
            return feedbackAnalysisCategorySet;
        }

        public void setFeedbackAnalysisCategorySet(String
feedbackAnalysisCategorySet) {
            this.feedbackAnalysisCategorySet = feedbackAnalysisCategorySet;
        }

        public List<String> getCategories() {
            return categories;
        }

        public void setCategories(List<String> categories) {
            this.categories = categories;
        }

        public List<Integer> getCounts() {
            return counts;
        }

        public void setCounts(List<Integer> counts) {
            this.counts = counts;
        }

    }

    @Inject
    @MessageBundle
    private transient ResourceBundle messages;

    private static final long serialVersionUID = 1L;

    private List<FeedbackAnalysisCategorySetEntity> categorySetsInUse;

    private FeedbackAnalysisEntity feedbackAnalysis;

    private List<BarChartModel> barModels;
```

```
    private List<PieChartModel> pieModels;

    private List<TableInformation> tableInformations;

    private boolean renderPieChart = true;

    private boolean renderBarChart = true;

    private final String SAVETODATABASE = "save";

    private final String DOWNLOAD = "download";

    private final String EMAIL = "mail";

    private String emailAddress;

    private boolean analyzationSaved = false;

    private List<String> selectedSaveOperations;

    @Inject
    private FeedbackAnalysisManagedBean feedbackAnalysisManagedBean;

    @Inject
    private Mailer mailerEJB;

    public String getEmailAddress() {
        return emailAddress;
    }

    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public List<String> getSelectedSaveOperations() {
        return selectedSaveOperations;
    }

    public void setSelectedSaveOperations(List<String> selectedSaveOperations)
{
        this.selectedSaveOperations = selectedSaveOperations;
    }

    public boolean isRenderPieChart() {
        return renderPieChart;
    }

    public void setRenderPieChart(boolean renderPieChart) {
        this.renderPieChart = renderPieChart;
    }

    public boolean isRenderBarChart() {
        return renderBarChart;
    }

    public void setRenderBarChart(boolean renderBarChart) {
```

```
        this.renderBarChart = renderBarChart;
    }

    public List<TableInformation> getTableInformations() {
        return tableInformations;
    }

    public void setTableInformations(List<TableInformation> tableInformations)
{
        this.tableInformations = tableInformations;
    }

    public List<PieChartModel> getPieModels() {
        return pieModels;
    }

    public void setPieModels(List<PieChartModel> pieModels) {
        this.pieModels = pieModels;
    }

    public List<BarChartModel> getBarModels() {
        return barModels;
    }

    public void setBarModels(List<BarChartModel> barModels) {
        this.barModels = barModels;
    }

    public List<FeedbackAnalysisCategorySetEntity> getCategorySetsInUse() {
        return categorySetsInUse;
    }

    public void setCategorySetsInUse(List<FeedbackAnalysisCategorySetEntity>
categorySetsInUse) {
        this.categorySetsInUse = categorySetsInUse;
    }

    public FeedbackAnalysisEntity getFeedbackAnalysis() {
        return feedbackAnalysis;
    }

    public void setFeedbackAnalysis(FeedbackAnalysisEntity feedbackAnalysis) {
        this.feedbackAnalysis = feedbackAnalysis;
    }

    public FeedbackAnalysisSummaryManagedBean() {

    }

    /**
     * Shows a dialog to tell that the analysis has been saved
     */
    public void showObservationSavedMessage() {
        if (analyzationSaved) {
            FacesContext.getCurrentInstance().addMessage(null,
                    new FacesMessage(FacesMessage.SEVERITY_INFO,
```

```
messages.getString("asum_analyzationSaved"), ""));
            analyzationSaved = false;
        }
    }

    /**
     * Tells whether the given keystring is in selectedSaveOperations.
     *
     * @param saveOperation
     *            The keystring of the selected save operation
     * @return whether the given operation is selected
     */
    public boolean isSelected(String saveOperation) {
        for (String s : selectedSaveOperations)
            if (s.contentEquals(saveOperation))
                return true;
        return false;
    }

    /**
     * Sends the email with the given files as an attachment
     *
     * @param files
     *            the files to be sent as an attachment
     */
    private void mail(List<File> files) {
        File[] filesArray = files.toArray(new File[files.size()]);
        String[] recipients = { emailAddress };

        mailerEJB.sendEmailWithAttachment(recipients, "Analysis results from
Moveatis",
                "Analysis results from Moveatis", filesArray);

        analyzationSaved = false;
    }

    /**
     * Does the selected save operations, saves to database, sends email or
     * downloads the csv
     *
     * @throws IOException
     */
    public void save() throws IOException {
        List<File> files = new ArrayList<>();
        String fileName = feedbackAnalysis.getAnalysisName();
        fileName = convertToFilename(fileName);

        if (isSelected(SAVETODATABASE)) {
            feedbackAnalysisManagedBean.saveFeedbackAnalysis();
            analyzationSaved = true;
        }

        if (isSelected(EMAIL)) {
            files.add(createCSV(fileName));
            files.add(DownloadTools.getImageFromByteArr(fileName,
feedbackAnalysisManagedBean.getReportImage()));
```

```
            files.add(DownloadTools.getImageFromByteArr(fileName,
feedbackAnalysisManagedBean.getPieImage()));
            files.add(DownloadTools.getImageFromByteArr(fileName,
feedbackAnalysisManagedBean.getBarImage()));
            files.add(DownloadTools.getImageFromByteArr(fileName,
feedbackAnalysisManagedBean.getTableImage()));

            mail(files);
            analyzationSaved = true;
        }

        if (isSelected(DOWNLOAD)) {
            DownloadTools.downloadCSV(getCSVData().toString(), fileName);
            analyzationSaved = true;
        }
        for (File file : files)
            file.delete();
    }

    /**
     * Downloads the right image based on the given keystring.
     *
     * @param whichFile
     *            the keystring that tells which image to download
     */
    public void downloadImage(String whichFile) {
        byte[] raw_img = null;
        String fileName = feedbackAnalysis.getAnalysisName();
        fileName = convertToFilename(fileName);
        if (whichFile.contentEquals("pie"))
            raw_img = feedbackAnalysisManagedBean.getPieImage();
        if (whichFile.contentEquals("bar"))
            raw_img = feedbackAnalysisManagedBean.getBarImage();
        if (whichFile.contentEquals("table"))
            raw_img = feedbackAnalysisManagedBean.getTableImage();
        if (raw_img == null)
            return;
        File img = DownloadTools.getImageFromByteArr(fileName + "_" + whichFile
+ "_", raw_img);
        DownloadTools.downloadFile(img, "image/png",
                img.getName().substring(0, img.getName().lastIndexOf("_")) +
".png");
        img.delete();
        analyzationSaved = true;
    }

    /**
     * File name converter.
     */
    private static String convertToFilename(String s) {
        if (s == null || s.isEmpty()) {
            return "unnamed";
        }
        return s.replaceAll("[^a-zA-Z0-9_]", "_");
    }
```

```
    /**
     * Creates a csv-file from the summary and report page information
     *
     * @param fileName
     *            the name with which the file should be made
     * @return
     */
    private File createCSV(String fileName) {
        StringBuilder sb = getCSVData();

        BufferedWriter writer = null;
        File csvFile = null;
        try {
            csvFile = File.createTempFile(fileName, ".csv");
            writer = new BufferedWriter(new FileWriter(csvFile));
            writer.write(sb.toString());
            writer.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return csvFile;
    }

    /**
     * Builds the csv out of the summary page information and appends the
     * information from the report page to it
     *
     * @return the csv-data in a StringBuilder
     */
    private StringBuilder getCSVData() {
        StringBuilder sb = new StringBuilder();

        sb.append("Name, " +
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity().getAnalysisName() +
"\n");
        sb.append("Target, " +
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity().getTargetOfAnalysis()
+ "\n");
        sb.append("Description, " +
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity().getDescription() +
"\n");
        sb.append("\n\n");

        for (TableInformation ti : tableInformations) {
            sb.append(ti.feedbackAnalysisCategorySet);
            sb.append(", n");
            sb.append(", %");
            sb.append("\n");
            for (int i = 0; i < ti.categories.size(); i++) {
                sb.append(ti.categories.get(i));
                sb.append(", ");
                sb.append(ti.counts.get(i).toString());
                sb.append(", ");
                sb.append(getPercentageAsString(ti.counts.get(i)) + "%");
                sb.append("\n");
            }
```

```
            sb.append("\n");
        }
        sb.append(feedbackAnalysisManagedBean.getReportCSV());
        return sb;
    }

    /**
     * calls the initModels function to build the summary table and the charts
     */
    @PostConstruct
    public void init() {
        initSummary();
        selectedSaveOperations = new ArrayList<>();
    }

    /**
     * Returns the counted percentage as a formatted string
     *
     * @param count
     *            number of records with a certain category selected
     * @return formatted string containing the percentage
     */
    public String getPercentageAsString(int count) {
        Locale locale = new Locale("en", "UK");
        String pattern = "##.##";
        DecimalFormat df = (DecimalFormat)
NumberFormat.getNumberInstance(locale);
        df.applyPattern(pattern);
        return df.format(countPercentage(count));

        // DecimalFormat df = new DecimalFormat("#.#");
        // return df.format(countPercentage(count));
    }

    /**
     * Counts the percentage of the given count out of the amount of records in
the
     * analysis
     *
     * @param count
     *            number of records with a certain category selected
     * @return The percentage of the given count out of the amount of records
     */
    private double countPercentage(int count) {
        return 100 * (double) count / (double)
feedbackAnalysis.getRecords().size();
    }

    /**
     * Gets the feedback analysis from the feedbackanalyzatinomanagedbean and
builds
     * the summary table and the charts based on the information contained in it
     */
    private void initSummary() {
        List<FeedbackAnalysisCategoryEntity> allSelectedCategories = new
ArrayList<FeedbackAnalysisCategoryEntity>();
```

```
        feedbackAnalysis =
feedbackAnalysisManagedBean.getFeedbackAnalysisEntity();
        categorySetsInUse =
feedbackAnalysisManagedBean.getFeedbackAnalysisCategorySetsInUse();
        int numberOfRecords = feedbackAnalysis.getRecords().size();
        for (FeedbackAnalysisRecordEntity record :
feedbackAnalysis.getRecords()) {
            allSelectedCategories.addAll(record.getSelectedCategories());
        }

        List<BarChartModel> barModels = new ArrayList<BarChartModel>();
        List<PieChartModel> pieModels = new ArrayList<PieChartModel>();
        List<TableInformation> tableInformations = new
ArrayList<TableInformation>();

        for (FeedbackAnalysisCategorySetEntity catSet : categorySetsInUse) {
            BarChartModel barModel = new BarChartModel();
            PieChartModel pieModel = new PieChartModel();
            TableInformation tableInformation = new
TableInformation(catSet.getLabel());
            int fullcount = 0;
            for (AbstractCategoryEntity cat :
catSet.getCategoryEntitys().values()) {
                ChartSeries categorySetChartSeries = new ChartSeries();
                categorySetChartSeries.setLabel(cat.getLabel().getText());
                int count = 0;
                // Comparison by category name and categoryset-name, because if
the analysis
                // hasn't yet been saved to the database the ID is null
                // categoryset-category pairs have to be unique
                for (FeedbackAnalysisCategoryEntity cat_comp :
allSelectedCategories)
                    if
(cat.getLabel().getText().contentEquals(cat_comp.getLabel().getText())
                            &&
catSet.getLabel().contentEquals(cat_comp.getCategorySet().getLabel()))
                        count++;
                fullcount += count;

                categorySetChartSeries.setLabel(cat.getLabel().getText() +",
"+getPercentageAsString(count)+"%");
                pieModel.set(cat.getLabel().getText() +",
"+getPercentageAsString(count)+"%", countPercentage(count));

                categorySetChartSeries.set("", countPercentage(count));
                barModel.addSeries(categorySetChartSeries);


    tableInformation.addCategoryWithCount(cat.getLabel().getText(), count);
            }
            if (numberOfRecords > fullcount) {
                int count=numberOfRecords-fullcount;
                ChartSeries empty = new ChartSeries();
                empty.setLabel("------"+",
"+getPercentageAsString(count)+"%");
                empty.set(catSet.getLabel() , countPercentage(count));
```

```
                barModel.addSeries(empty);
                pieModel.set("------"+", "+getPercentageAsString(count)+"%",
countPercentage(count));
                tableInformation.addCategoryWithCount("------", count);
            }
            pieModel.setTitle(catSet.getLabel());
            pieModel.setLegendPlacement(OUTSIDE);
            pieModel.setLegendPosition("s");
            pieModel.setMouseoverHighlight(false);

            barModel.setBarWidth(50);
            barModel.setTitle(catSet.getLabel());
            barModel.setStacked(true);
            barModel.setLegendPlacement(OUTSIDE);
            barModel.setLegendPosition("s");
            barModel.setMouseoverHighlight(false);
            Axis yAxis = barModel.getAxis(AxisType.Y);
            yAxis.setMin(0);
            yAxis.setTickFormat("%3d");
            yAxis.setTickInterval("10");
            yAxis.setMax(100);
            if (catSet != categorySetsInUse.get(0))
                barModel.setExtender("chartExtenderHideTicks");
            else
                barModel.setExtender("chartExtender");

            barModels.add(barModel);
            pieModels.add(pieModel);
            tableInformations.add(tableInformation);
        }

        this.barModels = barModels;
        this.pieModels = pieModels;
        this.tableInformations = tableInformations;
    }

    /**
     * Counts the maximum number of categories in a categoryset.
     *
     * @return The maximum number of categories in a categoryset.
     */
    public int countMaxCategories() {
        int max = 0;
        for (FeedbackAnalysisCategorySetEntity catSet : categorySetsInUse)
            if (catSet.getCategoryEntitys().size() > max)
                max = catSet.getCategoryEntitys().size();
        return max + 1;
    }

}
```

# src/main/java/com/moveatis/records/FeedbackAnalysisRecordBean.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.records;

import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

import com.moveatis.abstracts.AbstractBean;
import com.moveatis.interfaces.FeedbackAnalysisRecord;

/**
 * Controls the database connection for the feedbackanalysisrecords
 *
 * @author Visa Nykänen
 */
@Stateless
public class FeedbackAnalysisRecordBean extends
AbstractBean<FeedbackAnalysisRecordEntity>
        implements FeedbackAnalysisRecord {

    @PersistenceContext(unitName = "MOVEATIS_PERSISTENCE")
    private EntityManager em;
```

```
    @Override
    protected EntityManager getEntityManager() {
        return em;
    }

    public FeedbackAnalysisRecordBean() {
        super(FeedbackAnalysisRecordEntity.class);
    }
}
```

# src/main/java/com/moveatis/records/FeedbackAnalysisRecordEntity.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.records;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.JoinColumn;
```

```
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

import com.moveatis.abstracts.AbstractRecordEntity;
import com.moveatis.feedbackanalysis.FeedbackAnalysisEntity;
import com.moveatis.feedbackanalysiscategory.FeedbackAnalysisCategoryEntity;

/**
 * The entity for feedbackanalysis records, corresponds to the
 * feedbackanalysisrecord table in the database
 *
 * @author Visa Nykänen
 */
@Entity
@Table(name = "FEEDBACKANALYSISRECORD")
public class FeedbackAnalysisRecordEntity extends AbstractRecordEntity {

    @ManyToOne
    private FeedbackAnalysisEntity feedbackAnalysis;

    /**
     * The categories selected in this record
     */
    @ManyToMany
    @JoinTable(name = "FeedbackAnalysisRecordSelectedCategories", joinColumns =
@JoinColumn(name = "record_id"), inverseJoinColumns = @JoinColumn(name =
"category_id"))
    private List<FeedbackAnalysisCategoryEntity> selectedCategories;

    /**
     * Maintains the order of the records within an analysis
     */
    private Integer orderNumber;

    public Integer getOrderNumber() {
        return orderNumber;
    }

    public void setOrderNumber(Integer orderNumber) {
        this.orderNumber = orderNumber;
    }

    /**
     * Adds the given category to this record, makes sure that referential
integrity
     * is maintained by also editing the recordscontainingthiscategory-list of
the
     * category to be added
     *
     * @param category
     *            the category to be added
     */
    public void addSelectedCategory(FeedbackAnalysisCategoryEntity category) {
        selectedCategories.add(category);
```

```
        if (category.getRecordsContainingThisFeedbackAnalysisCategory() ==
null)
            category.setRecordsContainingThisFeedbackAnalysisCategory(new
ArrayList<FeedbackAnalysisRecordEntity>());

    category.getRecordsContainingThisFeedbackAnalysisCategory().add(this);
    }

    public void removeSelectedCategory(FeedbackAnalysisCategoryEntity category)
{
        selectedCategories.remove(category);

    category.getRecordsContainingThisFeedbackAnalysisCategory().remove(this);
    }

    public List<FeedbackAnalysisCategoryEntity> getSelectedCategories() {
        return selectedCategories;
    }

    public void setSelectedCategories(List<FeedbackAnalysisCategoryEntity>
selectedCategories) {
        this.selectedCategories = new
ArrayList<FeedbackAnalysisCategoryEntity>();
        for (FeedbackAnalysisCategoryEntity selectedCategory :
selectedCategories)
            addSelectedCategory(selectedCategory);
    }

    public FeedbackAnalysisEntity getFeedbackAnalysis() {
        return feedbackAnalysis;
    }

    public void setFeedbackAnalysis(FeedbackAnalysisEntity feedbackAnalysis) {
        this.feedbackAnalysis = feedbackAnalysis;
    }
}
```

# src/main/java/com/moveatis/restful/DataReceiver.java

```
/*
 * Copyright (c) 2016, Jarmo Juujärvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Visa Nykänen, Tuomas Moisio, Petra Puumala, Karoliina
Lappalainen
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
```

```
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */
package com.moveatis.restful;

import java.io.IOException;
import java.io.Serializable;

import javax.inject.Inject;
import javax.servlet.http.HttpServletRequest;
import javax.ws.rs.Consumes;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Context;
import javax.ws.rs.core.MediaType;

import com.moveatis.managedbeans.FeedbackAnalysisManagedBean;
import com.moveatis.managedbeans.ObservationManagedBean;

/**
 * Receives the images and csv-data created on the client side.
 *
 * @author Visa Nykänen
 */
@Path("/summary")
public class DataReceiver implements Serializable {

    private static final long serialVersionUID = 1L;

    @Inject
    private ObservationManagedBean observationManagedBean;

    @Inject
    private FeedbackAnalysisManagedBean feedbackAnalysisManagedBean;

    @Context
    private HttpServletRequest httpRequest;

    /**
```

```
     * Receives base64-encoded image file converts it to a byte array and stores
it
     * in the session.
     *
     * @param data
     *            The base64-encoded png-file
     * @throws IOException
     */
    @POST
    @Path("image")
    @Consumes(MediaType.TEXT_PLAIN)
    public void receiveImage(String data) throws IOException {
        String[] info = data.split(",");
        if (info.length != 3)
            return;
        String base64Image = info[2];
        byte[] img =
javax.xml.bind.DatatypeConverter.parseBase64Binary(base64Image);
        if (info[0].equals("obsimg"))
            observationManagedBean.setImage(img);
        if (info[0].equals("analpie"))
            feedbackAnalysisManagedBean.setPieImage(img);
        if (info[0].equals("analbar"))
            feedbackAnalysisManagedBean.setBarImage(img);
        if (info[0].equals("analtable"))
            feedbackAnalysisManagedBean.setTableImage(img);
        if (info[0].equals("reporttable"))
            feedbackAnalysisManagedBean.setReportImage(img);
    }

    /**
     * Receives csv-data as a string
     *
     * @param data
     *            the csv-data
     */
    @POST
    @Path("csv")
    @Consumes(MediaType.TEXT_PLAIN)
    public void receiveCSV(String data) {
        feedbackAnalysisManagedBean.setReportCSV(data);
    }

}
```

# src/main/webapp/META-INF/resources/css/analysiscategoryselection.css

```
/*
 * Copyright (c) 2016, Jarmo Juujarvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Karoliina Lappalainen, Tuomas Moisio, Visa Nykanen, Petra
Puumala
 * All rights reserved.
```

```
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
    last modified: 22.5.2019 by Petra Puumala
    Author     : Ilari Paananen
*/

#mainPage {
    text-align: center;
    margin-bottom: 1em;
    clear: both;
}

#container {
    width: 20em;
    display: inline-block;
    text-align: left;
}

#continue-div {
    margin-top: 1em;
    text-align: right;
}

.helpText {
    font-size: 1.15em;
}

h1{
```

```
    padding-bottom: 1em;
}

/* categorysets and categories */

.category-set-list {
    margin-bottom: 0.5em;
}

.category-set-list-last {
    margin-bottom: 2em;
}

.category-set-list-label {
    display: inline-block;
    width: 13.75em;
    vertical-align: middle;
}

.category-set-list-select {
    width: 10.625em;
    vertical-align: middle;
    margin-right: 1em;
}

.category-set-list-input {
    width: 12.313em;
    vertical-align: middle;
    margin-right: 1em;
}

.category-set {
    width: 14em;
    display: inline-block;
    vertical-align: top;
    margin: 0 0.625em 1em 0;
    background: rgba(85, 102, 119, 0.1);
    border-radius: 0.625em;
    padding: 0.625em;
}

.category-set h2 {
    margin-top: 0;
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
    font-size: 1.3em;
}

.category-set-remove-button {
    float: right;
    font-size: 150%;
}

.no-categories-text {
    margin-bottom: 0.5em;
```

```
}

.category {
    margin-bottom: 0.5em;
    white-space: nowrap;
}

.ui-chkbox-box {
    vertical-align: middle;
    margin-right: 0.5em;
}

.category .ui-chkbox-label {
    vertical-align: middle;
    font-size: 1.2em;
    display: inline;
}

.category-type-button {
    text-align: center;
    width: 4.5em;
    margin-right: 0.5em;
}

.category-type-button .ui-button-text {
    padding-left: 0;
    padding-right: 0;
}

.category-text {
    width: 75%;
    margin-right: 0.5em;
}

.category-remove-button {
    width: 2em;
}

.category .ui-message-error {
    background-color: transparent;
    border: 0;
}

.new-category-button {
    width: 100%;
    text-align: left;
}

/* icons */

.fa-remove {
    color: black;
}

.ui-icon, .ui-widget-content .ui-icon {
    background-image:
```

```
url("http://download.jqueryui.com/themeroller/images/ui-icons_FFFFFF_256x240.
png");
}

.ui-state-hover .ui-icon, .ui-state-focus .ui-icon, .ui-button:hover .ui-icon,
.ui-button:focus .ui-icon {
    background-image:
url("http://download.jqueryui.com/themeroller/images/ui-icons_FFFFFF_256x240.
png");
}

.ui-state-active .ui-icon, .ui-button:active .ui-icon {
    background-image:
url("http://download.jqueryui.com/themeroller/images/ui-icons_FFFFFF_256x240.
png");
}

.ui-state-highlight .ui-icon, .ui-button .ui-state-highlight.ui-icon {
    background-image:
url("http://download.jqueryui.com/themeroller/images/ui-icons_FFFFFF_256x240.
png");
}

.ui-button .ui-icon {
    background-image:
url("http://download.jqueryui.com/themeroller/images/ui-icons_FFFFFF_256x240.
png");
}

/* dialogs */

.ui-dialog-titlebar {
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    color: white;
    text-shadow: none;
}

.ui-confirm-dialog {
    border-left: 0.063em solid white;
    border-top: 0.063em solid white;
}

.ui-dialog-titlebar-close:hover, .ui-dialog-titlebar-close:focus {
    background-color: transparent;
    border: none;
    background-image: none;
    box-shadow: none;
}

.ui-dialog .ui-button {
    border: none;
    text-shadow: none;
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    cursor: pointer;
```

```
    color: white;
    box-shadow: none;
}

.ui-dialog .ui-button .ui-icon {
    color: black;
}

.ui-growl
.ui-icon{width:16px;height:16px;background-image:url("/javax.faces.resource/i
mages/ui-icons_616161_256x240.png.xhtml?ln=primefaces-aristo");}

.ui-dialog-titlebar-close:hover, .ui-dialog-titlebar-close:focus {
    background-color: transparent;
    border: none;
    background-image: none;
    box-shadow: none;
}

.ui-dialog .ui-button {
    border: none;
    text-shadow: none;
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    cursor: pointer;
    color: white;
    box-shadow: none;
}

.ui-dialog .ui-button .ui-icon {
    color: white;
}

/* buttons */

.ui-button {
    border: none;
    text-shadow: none;
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    cursor: pointer;
    color: white;
    box-shadow: none;
}

.ui-button .fa-remove {
    color: white;
}

.ui-button.ui-state-active {
    background-color: #AF2A01;
    background: linear-gradient(0deg, #007096, #007AA3);
}

/* responsivity */
```

```
@media screen and (min-width: 640px) {
    #container {
        width: 40em;
    }
}

@media screen and (min-width: 960px) {
    #container {
        width: 60em;
    }
}

@media screen and (max-width: 639px) {
    .category-set-list .ui-button{
        display: block;
        margin: auto;
        margin-top: 0.5em;

    }

    #container {
        width: 100%;
        text-align: center;
    }
}
```

# src/main/webapp/META-INF/resources/css/analysisrecordtable.css

```
/*
 * Copyright (c) 2016, Jarmo Juujarvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Karoliina Lappalainen, Tuomas Moisio, Visa Nykanen, Petra
Puumala
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
```

```
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
    last modified: 17.5.2019 by Petra Puumala
    Author     : Ilari Paananen
*/
html {
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
}

#mainPage {
    width: 100%;
    margin: auto;
    padding: 0;
    display: inline-block;
    text-align: center;
}

.html2canvas-container {
    width: 120em;
}

#container {
    margin: 1em;
    display: inline-block;
    font-size: 1.1em;
    padding: .5em .5em;
    background: rgba(85, 102, 119, 0.1);
    border-radius: 0.625em 0.625em 0.625em 0.625em;
    position: relative;
    z-index: 2;
    padding-bottom: 0em;
}

/* entries table */

#entries td:nth-child(1), #entries th:nth-child(1) {
    text-align: center;

}

#entries td:nth-child(2), #entries th:nth-child(2) {
    text-align: left;
    word-break: break-word;
}
```

```
#entries th:last-child, #entries td:last-child {
    border-right: 0em !important;
}

#entries {
    font-size: 1.0em;
}

#entries th {
    background: #E0E1E3;
    border: 0px;
    box-shadow: none;
    border-right: 0.063em solid #005977;
    color: black;
    hyphens: auto;
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: normal;
}

#entries td {
    border: 0em;
    border-right: 0.063em solid #005977;
    box-shadow: none;
    color: black;
    hyphens: auto;
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: normal;
}

#entries tr {
    border: 0em;
    box-shadow: none;
}

#entries tbody {
    border: 0em;
    box-shadow: none;
}

.ui-datatable-odd {
    background: #E0E1E3;
}

.ui-datatable-even {
    background: #D0D3D7;
}

#entries .ui-icon {
    border-radius: 0;
    font-size: 1em;
    height: 2em;
    padding-bottom: 1em;
}
```

```
/* buttons */

#downloadImage {
    text-align: center;
}

#continueToSummary {
    padding-right: 0.5em;
    text-align: right;
}

#entries td > span > .ui-button {
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    color: white;
    border: 0;
    text-shadow: none;
}

.ui-button {
    width: auto;
    margin: .6em 0;
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    cursor: pointer;
    color: white;
    border: 0;
    text-shadow: none;
}

#entries .ui-button.ui-state-active {
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
}

.ui-confirm-dialog .ui-button {
    margin-right: 0.5em!important;
}

#entries button {
    width: 2em;
    height: 2em;
    overflow: hidden;
}

#reportButtons {
    text-align: right;
    margin-right: 0.6em;
}

/* responsivity */

@media screen and (max-width: 800px) {
    .entries_dt .ui-datatable-data td .ui-column-title {
        display: none;
```

```
    }
    #entries table{width:100%!important;} /* Width has been defined in xhtml, so
this must be important*/
    .entries_dt thead th, .entries_dt tfoot td {
        display: none;
    }
    .entries_dt .ui-datatable-data td {
        text-align: left;
        display: flex;
        border: 0em none;
        -webkit-box-sizing: border-box;
        -moz-box-sizing: border-box;
        box-sizing: border-box;
        float: left;
        clear: left;
    }
    .entries_dt .ui-datatable-data.ui-widget-content {
        border: 0em none;
    }
    .entries_dt .ui-datatable-data tr.ui-widget-content {
        border-left: 0em none;
        border-right: 0em none;
    }
    .entries_dt .ui-datatable-data td .ui-column-title {
        padding: .4em;
        display: inline-block;
        margin: -.4em 1em -.4em -.4em;
    }
    #container {
        width: 90%;
    }
    #entries {
        margin-top: 0.5em;
        margin-left: 0.5em;
        margin-right: 0.5em;
        font-size: 1em;
    }
    #entries .ui-column-title {
        font-weight: bold;
        display: inline;
    }
    #entries td:last-child .ui-column-title:after {
        font-weight: bold;
        content: "";
    }
    #entries td:nth-last-child(2) .ui-column-title:after {
        font-weight: bold;
        content: "";
    }
    #entries td:last-child .ui-column-title {
        display: none;
    }
    #entries td:nth-last-child(2) .ui-column-title {
        display: none;
    }
    #entries_data td:nth-last-child(2) {
```

```
        width: auto;
        display: inline-block;
        float: right;
    }
    #entries td:last-child {
        width: auto;
        display: inline-block;
        float: right;
    }
    #entries td:nth-child(2) .ui-column-title {
        display: block;
    }
    #entries .ui-column-title:after {
        font-weight: bold;
        content: ":";
    }
    #entries td:first-child .ui-column-title {
        display: inline;
    }
    #entries td .ui-column-title {
        display: inline-block;
        overflow: hidden;
        hyphens: none;
        text-overflow: ellipsis;
    }
    #entries td {
        border: 0em;
        border-right: 0em;
        box-shadow: none;
        color: black;
        hyphens: none;
        white-space: normal;
        padding: 0;
        border-right: 0em;
    }
    #entries td:nth-child(1), #entries th:nth-child(1), #entries
td:nth-child(2), #entries th:nth-child(2) {
        text-align: left;
    }
    #entries tr {
        border: 0em;
        box-shadow: none;
    }
    #entries .tContent {
        display: inline-block;
    }
    #reportButtons {
        text-align: right;
        margin-right: 0em;
    }
}

@media screen and (max-width: 400px) {
    #entries {
        font-size: 0.8em;
    }
```

```
    #container {
        width: 90%;
    }
}
``````

src/main/webapp/META-INF/resources/css/analyzer.css

``````
/*
 * Copyright (c) 2016, Jarmo Juujarvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Karoliina Lappalainen, Tuomas Moisio, Visa Nykanen, Petra
Puumala
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
        last modified: 25.5.2019 by Petra Puumala
        Author     : Ilari Paananen
*/

body {
    text-align: center;
    /* Disable text selection.
                Source:
http://stackoverflow.com/questions/826782/css-rule-to-disable-text-selection-
highlighting */
    -webkit-touch-callout: none;
```

```
    /* iOS Safari */
    -webkit-user-select: none;
    /* Chrome/Safari/Opera */
    -khtml-user-select: none;
    /* Konqueror */
    -moz-user-select: none;
    /* Firefox */
    -ms-user-select: none;
    /* IE/Edge */
    user-select: none;
    /* non-prefixed version, currently
     not supported by any browser */
}

#mainPage {
    width: 100%;
    margin: auto;
    padding: 0;
    display: inline-block;
    text-align: center;
}

#index {
    padding-top: 0.625em;
    padding-bottom: 0.625em;
}

#naviHead, #footer {
    font-size: 1em;
}

#texts {
    display: inline-block;
    width: 25%;
    hyphens: auto;
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: nowrap;
    padding-top: 0.5em;
    vertical-align: top;
    font-size: 1.1em;
}

#content {
    width: 60%;
    margin: 0;
    padding: 0;
    display: inline-block;
}

#content .ui-state-active {
    background: #AF2A01;
}

#textArea {
    text-align: center;
```

```
}

#textArea textarea {
    max-width: 100%;
    min-width: 98%;
    border-radius: 0.5em;
    margin-bottom: 0.1em;
}

#categoryContent {
    max-width: 100%;
    list-style-type: none;
    background: rgba(85, 102, 119, 0.1);
    padding: 0.5em;
    border-radius: 0.5em;
    position: relative;
}

#basicContent {
    width: 60%;
    margin: 0;
    padding: 0 0 1em 0;
    display: inline-block;
}

#categoryContent {
    margin-bottom: 0.625em;
    text-align: left;
    padding-left: 2em;
}

/* timer and timer buttons */
#timer_div {
    padding-top: 1em;
    padding-bottom: 1em;
}

#timer_div .ui-button.ui-state-active {
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
}

#timer_div .ui-icon {
    border-radius: 0;
    font-size: 1em;
}

#timer_span {
    font-size: 2em;
    display: inline-block;
    position: relative;
    vertical-align: middle;
}

#pause_span {
    padding-left: 0.8em;
```

```
    vertical-align: middle;
    display: inline-block;
}

#timestamp_div {
    max-width: 100%;
    list-style-type: none;
    background: rgba(85, 102, 119, 0.1);
    padding: 0.5em;
    border-radius: 0.5em;
    position: relative;
    text-align: center;
    margin-bottom: 0.5em;
}

#timestamp_span {
    padding-left: 2em;
    font-weight: bold;
    text-decoration:underline;
}

/* page number */

#pageNumber {
    display: inline-block;
    width: 3em;
}

/* buttons */

#content button {
    padding: 0.5em;
    border: 0;
    font-size: 1em;
    display: inline-block;
    min-width: 2.5em;
    max-width: 5em;
}

#buttons {
    display: inline-block;
    margin-right: 0;
    margin-left: auto;
    width: 74%;
}

.ui-button {
    display: inline-block;
    border: none;
    text-shadow: none;
    margin: 0.2em 0.2em;
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    cursor: pointer;
    color: white;
    box-shadow: none;
```

```
    max-width: 12em;
}

.ui-button-text {
    text-overflow: ellipsis;
    overflow: hidden;
    white-space: nowrap;
}

.ui-button.ui-state-active {
    background: #AF2A01;
}

/* timeslider */
#timeSlider\:timeOutput {
    color: white;
    margin: 1em;
}

#timeSlider > div {
    margin: 0.5em 0;
}


/* responsivity */

@media screen and (max-width: 1050px) {
    #texts {
        display: inline-block;
        width: 100%;
        hyphens: auto;
        text-overflow: ellipsis;
        overflow: hidden;
        white-space: nowrap;
        padding-top: 0.5em;
        vertical-align: top;
    }
    #buttons {
        display: inline-block;
        margin-right: 0;
        margin-left: auto;
        width: 100%;
    }
}

@media screen and (max-width: 600px) {
    #index {
        width: 100%;
        display: flex;
        flex-flow: row wrap;
        overflow: hidden;
    }
    #pageNumber {
        display: block;
        margin: auto;
        order: 1;
```

```
        width: 100%;
    }
    .arrows {
        order: 2;
        margin: auto;
    }
    #texts {
        display: inline-block;
        width: 100%;
        hyphens: auto;
        text-overflow: ellipsis;
        overflow: hidden;
        white-space: nowrap;
        padding-top: 0.5em;
        vertical-align: top;
    }
    #buttons {
        display: inline-block;
        margin-right: 0;
        margin-left: auto;
        width: 100%;
    }
}

@media screen and (max-width: 480px) {
    .arrows .ui-button {
        margin: 0;
        width: 2.6em;
    }
    #timestamp_span {
        white-space: nowrap;
        padding-left: 0;
    }
}

@media screen and (max-width: 400px) {
    .ui-button {
        display: inline-block;
        border: none;
        text-shadow: none;
        margin: 0.2em 0.2em;
        cursor: pointer;
        color: white;
        box-shadow: none;
        max-width: 8em;
    }
    #categoryContent {
        padding-left: 0.5em;
    }
    #basicControls .ui-button {
        max-width: 11em;
    }
}
``````

src/main/webapp/META-INF/resources/css/feedbackanalysissummary.css

``````
```
/*
 * Copyright (c) 2016, Jarmo Juujarvi, Sami Kallio, Kai Korhonen, Juha Moisio,
Ilari Paananen
 * Copyright (c) 2019, Karoliina Lappalainen, Tuomas Moisio, Visa Nykanen, Petra
Puumala
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     1. Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *
 *     2. Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *
 *     3. Neither the name of the copyright holder nor the names of its
 *       contributors may be used to endorse or promote products derived
 *       from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/*
    last modified: 22.5.2019 by Petra Puumala
    Author     : Ilari Paananen
*/

html {
    width: 100%;
    height: 100%;
    margin: 0;
    padding: 0;
}

.html2canvas-container {
    width: 120em;
    height: 187.5em;
}

#mainPage {
    width: 100%;
```
```

```
    margin: auto;
    padding: 0;
    display: inline-block;
    text-align: center;
}

#container {
    width: 90%;
    margin: 1em;
    display: inline-block;
    font-size: 1.1em;
    padding: .5em .5em;
    background-color: rgba(85, 102, 119, 0.1);
    border-radius: 0.625em 0.625em 0.625em 0.625em;
    position: relative;
    z-index: 2;
}

.summaryTable {
    width: 100%;
    border: 0;
    border-collapse: collapse;
    margin-top: 1em;
}

.tableheader {
    text-align: left;
    border-bottom: 0.1em solid #005977;
}

.ui-growl-message {
    padding-top: 2em;
}

.ui-growl-image-info {
    margin-top: 1.5em;
}

.tableheader, .cat {
    width: 40%;
}

.cat, .count {
    text-align: left;
    padding: 0.5em;
    border-bottom: 0.09em solid #005977;
}

.ui-button {
    width: auto;
    margin: .6em 0;
    background-color: #008AB9;
    background: linear-gradient(0deg, #008AB9, #10A1D1);
    cursor: pointer;
    color: white;
    border: 0;
```

```
    text-shadow: none;
}

/* charts */

#charts {
    padding-top: 1em;
    padding-bottom: 1em;
}

#bar_div {
    margin-top: 1em;
    background: white;
    display: inline-block;
    border: 0.063em solid black;
    text-align: left;
    white-space: nowrap;
    overflow-x: auto;
    min-width:8em;
    max-width: 90%;
}

#charts_div {
    text-align: left;
    padding-left: 2em;
}

#chartsHeader {
    font-weight: bold;
    margin-bottom: 0.3em;
}

.chart {
    display: inline-block;
    width: 9em;
}

.jqplot-title {
    white-space: nowrap;
    font-size: 1em;
    overflow: hidden;
    text-overflow: ellipsis;
}

.jqplot-highlighter-tooltip{
    background: white;
}

table.jqplot-table-legend td {
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
    max-width: 9em;
}

#piechart .jqplot-table-legend {
```

```
    left: 2em;
    width: 70%;
}

#piechart .jqplot-table-legend-swatch {
    width: 3%;
}

#piechart .jqplot-table-legend-label {
    width: 97%;
    text-align: left;
}

#pieModels canvas {
    width: 18.75em;
}

#saveForm {
    text-align: left;
}

/* button positioning */

.hiddenButtons {
    margin-left: 1.5em;
    margin-right: 1.5em;
    text-align: right;
}

.back, .end, .save {
    display: inline-block;
}

.back {
    float: left;
}

.save {
    margin-right: 1em;
}

/* responsivity */

@media ( max-width: 570px) {
    #container {
        width: 80%;
    }
    #bar_div {
        font-size: 0.8em;
        padding: 0.5em;
    }
    #hiddenButtons {
        margin-left: 0;
        margin-right: 0;
    }
    .back, .end, .save {
```

```
        display: inline-block;
    }
    .back {
        width: 100%;
        text-align: right;
    }
    .end {
        width: 100%;
        text-align: right;
        padding: 0;
    }
}

@media ( max-width: 750px) {
    .save {
        margin-right: 0;
    }
}

@media ( max-width: 400px) {
    #container {
        width: 90%;
        font-size: 0.8em;
    }
    #bar_div {
        font-size: 1em;
        padding: 3em;
        width: 70%;
    }
    #piechart .jqplot-table-legend-swatch {
        width: 3%;
    }
    #piechart .jqplot-table-legend-label {
        width: 97%;
        text-align: left;
    }
    #piechart {
        width: 80%;
    }
}
``````

src/main/webapp/META-INF/resources/js/analysisrecordtable.js

``````
/**
 * @fileOverview JavaScript methods for report view.
 * @module analysisrecordtable
 * @author Visa Nykänen
 * @author Tuomas Moisio
 */

var URI;
function save() {
    var checkBoxCsv = document.getElementById('saveForm:basic:1');
    var checkBoxImage = document.getElementById('saveForm:basic:0');
```

```
    var filename = document.getElementById('saveForm:input-name').value;
    if (filename === "") {
        return;
    }

    if (checkBoxImage.checked) {
        saveAsImage();
    }
    if (checkBoxCsv.checked) {
        setTimeout(function() {
            saveAsCsv();
        }, 100);
    }

}

$(document).ready(function() {
    sendImageAndCSV();
});

/**
 * Creates the csv using table2csv.js, sends the csv and the image of the report
page to the server via AJAX
 */
function sendImageAndCSV() {
    // makes sure the header isn't hidden
    let tmpclass = $('#entries').attr('class');
    $('#entries').attr('class', "");
    $('#entries table')
            .each(
                    function() {
                        let $table = $(this);
                        // selector-settings for table2csv explained:
                        // the two last columns are edit and remove, so we don't
                        // really want them in our csv
                        // if the page is too narrow the table puts the header
                        // into each of the cells so only spans with the class
                        // tContent should be printed into the csv
                        let csv = $table
                                .table2CSV({
                                    delivery : 'value',
                                    headerSelector :
'th:not(:nth-last-child(1)):not(:nth-last-child(2))',
                                    columnSelector :
'td:not(:nth-last-child(1)):not(:nth-last-child(2)) span.tContent'
                                });
                        sendData(csv, "csv")
                    });
    $('#entries').attr('class', tmpclass);
    html2canvas(document.getElementById('dataTableImage')).then(
            function(canvas) {
                URI = canvas.toDataURL();
                sendData("reporttable," + URI, "image")
            })
}
```

```
/**
 * Sends the given data to the given page through ajax
 *
 * @param URI
 *            the data to be sent
 * @param page
 *            the page to which the data should be sent, currently either csv or
 *            image
 */
function sendData(URI, page) {
    $.ajax({
        url : "../../webapi/summary/" + page,
        type : "POST",
        dataType : "text",
        contentType : "text/plain",
        cache : false,
        data : URI,
        success : function(data) {

        },
        error : function(xhr, status, error) {
            showError(msg.obs_errorCouldntSendData + ": " + error);
            this_.waiting = false;
        }
    });
}

function saveAsCsv() {
    document.getElementById('saveForm:csvButton').click();
}

function saveAsImage() {
    html2canvas(document.getElementById('dataTableImage')).then(
            function(canvas) {
                URI = canvas.toDataURL();
                var filename;
                var filenameRaw;

                try {
                    filenameRaw = document
                            .getElementById('saveForm:input-name').value;
                    if (filenameRaw === "") {
                        return;
                    }
                    filename = filenameRaw.replace(/\./g, '-');
                } catch (err) {
                    filename = 'summary.png';
                }
                var link = document.createElement('a');
                if (typeof link.download === 'string') {
                    link.href = URI;
                    link.download = filename;

                    // Firefox requires the link to be in the body
                    document.body.appendChild(link);
```

```
                    // simulate click
                    link.click();

                    // remove the link when done
                    document.body.removeChild(link);
                } else {
                    window.open(URI);
                }
            });
}
``````

src/main/webapp/META-INF/resources/js/feedbackanalysissummary.js

``````
/**
 * @fileOverview JavaScript methods for feedbackanalysis summary view.
 * @module feedbackanalysissummary
 * @author Tuomas Moisio
 * @author Visa Nykänen
 */
var URI;
var arr = [];
function save() {
    let checkBoxImage = document.getElementById('saveForm:basic:1');
    let checkBoxImage2 = document
            .getElementById('saveForm:anonymityUserBoxes:1');

    let filename = document.getElementById('saveForm:input-name').value;
    if (filename === "") {
        return;
    }
    if (checkBoxImage != null) {
        if (checkBoxImage.checked) {
            for (let i = 0; i < arr.length; i++) {
                saveAsImage(arr[i]);
            }
        }
    }
    if (checkBoxImage2 != null) {
        if (checkBoxImage2.checked) {
            for (let j = 0; j < arr.length; j++) {
                saveAsImage(arr[j]);
            }
        }
    }
}
function createImage() {
    exportChart2();
    html2canvas(document.getElementById('tableImage')).then(function(canvas) {
        let array = [];
        arr = array;
        arr.push(canvas.toDataURL());
    });

    if (document.getElementById('charts:barChart_input').checked) {
```

```
        html2canvas(document.getElementById('barimages')).then(
                function(canvas) {
                    arr.push(canvas.toDataURL());
                });
    }
    if (document.getElementById('charts:pieChart_input').checked) {
        html2canvas(document.getElementById('pieimages')).then(
                function(canvas) {
                    arr.push(canvas.toDataURL());
                });
    }
}

/**
 * Sends an image through ajax to the servlet that handles images.
 *
 * @param URI
 *            the base64-encoded image to be sent
 */
function sendImage(URI) {
    $.ajax({
        url : "../../webapi/summary/image",
        type : "POST",
        dataType : "text",
        contentType : "text/plain",
        cache : false,
        data : URI,
        success : function(data) {

        },
        error : function(xhr, status, error) {
            showError(msg.obs_errorCouldntSendData + ": " + error);
            this_.waiting = false;
        }
    });
}

/**
 * Sends all the images created on the summary-page to the servlet that handles
 * them
 */
function sendImages() {
    html2canvas(document.getElementById('tableImage')).then(function(canvas) {
        let URI = "analtable," + canvas.toDataURL();
        sendImage(URI)
    });
    try {
        html2canvas(document.getElementById('piechartimage')).then(
                function(canvas) {
                    let URI = "analpie," + canvas.toDataURL();
                    sendImage(URI)
                });
    } catch (err) {
    }

    try {
```

```
        html2canvas(document.getElementById('barchartimage')).then(
                function(canvas) {
                    let URI = "analbar," + canvas.toDataURL();
                    sendImage(URI)
                });
    } catch (err) {
    }
}

$(document).ready(function() {
    sendImages();
})

$(window).load(function() {
    $('td.jqplot-table-legend-label').each(function(index) {
        $(this).attr('title', $(this).text());
    });
    $(document).tooltip();
});

function saveAsImage(dataURL) {
    var filename;
    var filenameRaw;
    try {
        filenameRaw = document.getElementById('saveForm:input-name').value;
        if (filenameRaw === "") {
            return;
        }
        filename = filenameRaw.replace(/\./g, '-');
    } catch (err) {
        filename = 'summary.png';
    }
    var link = document.createElement('a');
    if (typeof link.download === 'string') {
        link.href = dataURL;
        link.download = filename;

        // Firefox requires the link to be in the body
        document.body.appendChild(link);

        // simulate click
        link.click();

        // remove the link when done
        document.body.removeChild(link);
    } else {
        window.open(URI);
    }
}

function exportChart2() {
    let count = document.getElementById('chartCount').innerHTML;
    for (let index = 0; index < count; index++) {
        let b = 'piechart' + index;
        let a = 'barchart' + index;
        let linebreak = document.createElement('br');
```

```
        if (document.getElementById('charts:pieChart_input').checked) {

    document.getElementById('pieimages').append(PF(b).exportAsImage());
        }

        if (document.getElementById('charts:barChart_input').checked) {

    document.getElementById('barimages').append(PF(a).exportAsImage());
        }
    }
}
```

# src/main/webapp/META-INF/resources/js/frontpage.js

```
/**
 * @fileOverview Javascript methods for the front page
 * @module frontpage
 * @author Visa Nykänen
 * @author Juha Moisio
 */
$(document).ready(function() {
    sendTimezone();
});

/**
 * Sends the timezone information to the server that stores it in a session.
 */
function sendTimezone(){
    $.ajax({
        url : "../../webapi/records/settimezone",
        type : "POST",
        dataType : "text",
        contentType : "application/json",
        cache : false,
        data : JSON.stringify({
            timeZoneOffsetInMs : getTimeZoneOffset(),
            daylightSavingInMs : getDaylightSaving(),
        }),
        success : function(data) {
        },
        error : function(xhr, status, error) {
        }
    });
}


/**
 * Gets the offset of the time zone in milliseconds (in JAVA format).
 */
function getTimeZoneOffset(){
    return -1 * 60 * 1000 * new Date().getTimezoneOffset();
}

/**
```

```
 * Gets the daylight saving time offset in milliseconds.
 */
function getDaylightSaving() {
    var now = new Date();
    var jan = new Date(now.getFullYear(), 0, 1);
    return (jan.getTimezoneOffset() - now.getTimezoneOffset()) * 60 * 1000;
}
```

``````

src/main/webapp/META-INF/resources/js/table2csv.js

``````
```
jQuery.fn.table2CSV = function(options) {
    var options = jQuery.extend({
        separator : ',',
        header : [],
        headerSelector : 'th',
        columnSelector : 'td',
        delivery : 'popup', // popup, value, download
        // filename: 'powered_by_sinri.csv', // filename to download
        transform_gt_lt : true
    // make &gt; and &lt; to > and <
    }, options);

    var csvData = [];
    var headerArr = [];
    var el = this;

    // header
    var numCols = options.header.length;
    var tmpRow = []; // construct header avalible array

    if (numCols > 0) {
        for (var i = 0; i < numCols; i++) {
            tmpRow[tmpRow.length] = formatData(options.header[i]);
        }
    } else {
        $(el).filter(':visible').find(options.headerSelector).each(function()
{
            if ($(this).css('display') != 'none')
                tmpRow[tmpRow.length] = formatData($(this).html());
        });
    }

    row2CSV(tmpRow);

    // actual data
    $(el).find('tr').each(
            function() {
                var tmpRow = [];
                $(this).filter(':visible').find(options.columnSelector).each(
                        function() {
                            if ($(this).css('display') != 'none')
                                tmpRow[tmpRow.length] = formatData($(this)
                                        .html());
```

```
                    });
                row2CSV(tmpRow);
            });
    if (options.delivery == 'popup') {
        var mydata = csvData.join('\n');
        if (options.transform_gt_lt) {
            mydata = sinri_recover_gt_and_lt(mydata);
        }
        return popup(mydata);
    } else if (options.delivery == 'download') {
        var mydata = csvData.join('\n');
        if (options.transform_gt_lt) {
            mydata = sinri_recover_gt_and_lt(mydata);
        }
        var url = 'data:text/csv;charset=utf8,' + encodeURIComponent(mydata);
        window.open(url);
        return true;
    } else {
        var mydata = csvData.join('\n');
        if (options.transform_gt_lt) {
            mydata = sinri_recover_gt_and_lt(mydata);
        }
        return mydata;
    }

    function sinri_recover_gt_and_lt(input) {
        var regexp = new RegExp(/&gt;/g);
        var input = input.replace(regexp, '>');
        var regexp = new RegExp(/&lt;/g);
        var input = input.replace(regexp, '<');
        return input;
    }

    function row2CSV(tmpRow) {
        var tmp = tmpRow.join(''); // to remove any blank rows
        // alert(tmp);
        if (tmpRow.length > 0 && tmp != '') {
            var mystr = tmpRow.join(options.separator);

            csvData[csvData.length] = mystr;
        }
    }
    function formatData(input) {
        // replace " with “
        var regexp = new RegExp(/["]/g);
        var output = input.replace(regexp, "“");
        // HTML
        var regexp = new RegExp(/\<[^\<]+\>/g);
        var output = output.replace(regexp, "");
        output = output.replace(/ /gi, ' '); // replace  
        if (output == "")
            return '';
        return '"' + output.trim() + '"';
    }
    function popup(data) {
        var generator = window.open('', 'csv', 'height=400,width=600');
```

```
        generator.document.write('<html><head><title>CSV</title>');
        generator.document.write('</head><body >');
        generator.document.write('<textArea cols=70 rows=15 wrap="off" >');
        generator.document.write(data);
        generator.document.write('</textArea>');
        generator.document.write('</body></html>');
        generator.document.close();
        return true;
    }
};
```

# src/main/webapp/app/activityselection/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:p="http://primefaces.org/ui"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<body>
    <ui:composition template="/WEB-INF/template.xhtml">
        <ui:define name="mainpage">
            <div>
                <nav> <h:form>
                    <h:panelGroup>
                        <div class="accessButtons" style="min-height: 200px;">
                            <p:commandButton value="#{msg.selection_analyze}"
                                style="width: 200px; margin-top: 15px"
class="coloredButton"
                                action="#{controlBean.redirectAnalyze}" />
                            <p:commandButton
value="#{msg.selection_observation}"
                                ajax="false" style="width: 200px; margin-top:
15px"
                                class="coloredButton"
                                action="#{controlBean.redirectObservation}" />
                        </div>
                    </h:panelGroup>
                </h:form> </nav>
                <h:panelGroup styleClass="welcomeContent">
                    <h:outputText escape="false"
value="#{msg.selection_introduction}" />
                    <h:outputLink value="#{msg.fp_infoLink}">
                        <h:outputText value="#{msg.fp_infoLinkText}"
                            style="text-decoration: underline" />
                    </h:outputLink>
                </h:panelGroup>
                <div id="picture2">
                    <p:graphicImage library="images" name="valintasivu.png" />
                </div>
            </div>
        </ui:define>
```

```
    </ui:composition>
</body>
</html>
```

# src/main/webapp/app/analysiscategoryselection/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://xmlns.jcp.org/jsf/core">
<h:body>
    <ui:composition template="/WEB-INF/template.xhtml">
        <ui:define name="pageTitle">#{msg.facs_title}</ui:define>
        <ui:define name="mainpage">
            <h:outputStylesheet library="css"
                name="analysiscategoryselection.css" />
            <div id="container">
                <h:panelGroup layout="block"
                    rendered="#{empty
feedbackAnalysisCategorySelectionManagedBean.eventGroup}">
                    <h1>#{msg.facs_heading}</h1>
                </h:panelGroup>
                <h:panelGroup layout="block"
                    rendered="#{not empty
feedbackAnalysisCategorySelectionManagedBean.eventGroup}">
                    <h1>
                        <h:outputFormat value="#{msg.facs_headingEvent}">
                            <f:param

    value="#{feedbackAnalysisCategorySelectionManagedBean.eventGroup.label}"
/>
                        </h:outputFormat>
                    </h1>
                </h:panelGroup>
                <h2>#{msg.facs_addedCategorySets}</h2>
                <h:form id="form" onkeypress="return captureEnter(event);">
                    <p:growl id="growl" showDetail="true" />
                    <p:confirmDialog global="true">
                        <p:commandButton value="#{msg.dialogConfirmYes}"
type="button"
                            styleClass="ui-confirmdialog-yes"
icon="ui-icon-check" />
                        <p:commandButton value="#{msg.dialogConfirmNo}"
type="button"
                            styleClass="ui-confirmdialog-no"
icon="ui-icon-close" />
                    </p:confirmDialog>
                    <h:outputText value="#{msg.facs_noCategorySetsText}"
                        rendered="#{empty
```

```
feedbackAnalysisCategorySelectionManagedBean.feedbackAnalysisCategorySetsInUs
e}" />
                    <h:panelGroup id="category-sets">
                        <ui:repeat

    value="#{feedbackAnalysisCategorySelectionManagedBean.feedbackAnalysisCat
egorySetsInUse.toArray()}"
                            var="categorySet" varStatus="status">
                            <h:panelGroup id="category-set"
                                styleClass="category-set
category-set-#{status.index}">
                                <p:commandLink
                                    styleClass="fa fa-remove
category-set-remove-button"

    action="#{feedbackAnalysisCategorySelectionManagedBean.removeCategorySet(
status.index)}"
                                    immediate="true" update="@form">
                                    <p:confirm
header="#{msg.dialogConfirmTitle}"

    message="#{msg.facs_confirmRemoveCategorySet}"
                                        icon="ui-icon-alert" />
                                </p:commandLink>
                                <h2>#{categorySet.label}</h2>
                                <h:panelGroup layout="block"
styleClass="no-categories-text"
                                    rendered="#{empty
categorySet.categoryEntitys}">
                                    <h:outputText
value="#{msg.facs_noCategoriesText}" />
                                </h:panelGroup>
                                <ui:repeat

    value="#{categorySet.categoryEntitys.values().toArray()}"
                                    var="category">
                                    <div class="category">
                                        <p:inputText
value="#{category.label.text}"
                                            id="category-name-input"
styleClass="category-text"

    validator="#{validationBean.validateShortString}"

    onblur="this.value=this.value.trim()">
                                            <p:ajax event="blur"
process="@this" update="@this form:growl" />
                                        </p:inputText>
                                        <p:watermark for="category-name-input"
value="#{msg.facs_categoryName}" />
                                        <p:commandButton icon="fa fa-remove"
                                            class="category-remove-button"
                                            immediate="true"

    action="#{feedbackAnalysisCategorySelectionManagedBean.removeCategoryFrom
CategorySet(categorySet,category)}"
```

```
                                                update="form:category-sets
form:continue-button">
                                            </p:commandButton>
                                        </div>
                                    </ui:repeat>
                                    <p:commandButton
value="#{msg.facs_newCategory}"
                                        icon="fa fa-plus"
styleClass="new-category-button"

    action="#{feedbackAnalysisCategorySelectionManagedBean.addNewCategoryToCa
tegorySet(categorySet)}"
                                        update="category-set form:continue-button
form:growl" process="@this category-set"

    oncomplete="focusCategory('.category-set-#{status.index}');" />
                            </h:panelGroup>
                        </ui:repeat>
                    </h:panelGroup>
                    <br /><br /><br />
                    <h2>#{msg.facs_addCategorySets}</h2>
                    <h:panelGroup layout="block"
styleClass="category-set-list">
                            <p:message id="new-category-set-error"
for="new-category-set" />
                            <p:outputLabel for="new-category-set"
                                value="#{msg.facs_newCategorySet}:"
                                styleClass="category-set-list-label" />

                            <p:inputText

    value="#{feedbackAnalysisCategorySelectionManagedBean.newFeedbackAnalysis
CategorySetName}"
                                id="new-category-set"
styleClass="category-set-list-input"
                                validator="#{validationBean.validateShortString}"
                                onblur="this.value=this.value.trim()" >
                                <p:ajax event="keyup" process="@this"
update="addCategorySetButton new-category-set-error"/>
                                </p:inputText>
                            <p:watermark for="new-category-set"
value="#{msg.facs_categorySetName}"/>
                            <p:commandButton id="addCategorySetButton"
value="#{msg.facs_addCategorySet}"
                                style="vertical-align: middle" process="@this
new-category-set"

    disabled="#{feedbackAnalysisCategorySelectionManagedBean.newFeedbackAnaly
sisCategorySetName==null||feedbackAnalysisCategorySelectionManagedBean.newFee
dbackAnalysisCategorySetName.isEmpty()}"

    action="#{feedbackAnalysisCategorySelectionManagedBean.addNewCategorySet(
)}"
                                update="@form" />
                    </h:panelGroup>
                    <h:panelGroup layout="block"
```

```
styleClass="category-set-list"
                        rendered="#{not empty
feedbackAnalysisCategorySelectionManagedBean.privateFeedbackAnalysisCategoryS
ets.toArray()}">
                        <p:outputLabel for="private-category-sets"
                            value="#{msg.facs_privateCategorySets}:"
                            styleClass="category-set-list-label" />
                        <p:selectOneMenu id="private-category-sets"

    value="#{feedbackAnalysisCategorySelectionManagedBean.selectedPrivateFeed
backAnalysisCategorySet}"
                            styleClass="category-set-list-select">
                            <f:selectItems

    value="#{feedbackAnalysisCategorySelectionManagedBean.privateFeedbackAnal
ysisCategorySets.toArray()}"
                                var="categorySet"
itemLabel="#{categorySet.label}"
                                itemValue="#{categorySet.id}" />
                        </p:selectOneMenu>
                        <p:commandButton value="#{msg.facs_addCategorySet}"
                            style="vertical-align: middle"

    action="#{feedbackAnalysisCategorySelectionManagedBean.addPrivateCategory
Set}"
                            update="@form" />
                    </h:panelGroup>
                    <h:panelGroup layout="block"
styleClass="category-set-list-last"
                        rendered="#{not empty
feedbackAnalysisCategorySelectionManagedBean.defaultFeedbackAnalysisCategoryS
ets.toArray()}">
                        <p:outputLabel for="default-category-sets"
                            value="#{msg.facs_defaultCategorySets}:"
                            styleClass="category-set-list-label" />
                        <p:selectOneMenu id="default-category-sets"

    value="#{feedbackAnalysisCategorySelectionManagedBean.selectedDefaultFeed
backAnalysisCategorySet}"
                            styleClass="category-set-list-select">
                            <f:selectItems

    value="#{feedbackAnalysisCategorySelectionManagedBean.defaultFeedbackAnal
ysisCategorySets.toArray()}"
                                var="categorySet"
itemLabel="#{categorySet.label}"
                                itemValue="#{categorySet.id}" />
                        </p:selectOneMenu>
                        <p:commandButton value="#{msg.facs_addCategorySet}"
                            style="vertical-align: middle"

    action="#{feedbackAnalysisCategorySelectionManagedBean.addDefaultCategory
Set}"
                            update="@form" />
                    </h:panelGroup>
                    <br />
```

```
                    <p:selectBooleanCheckbox

    value="#{feedbackAnalysisCategorySelectionManagedBean.isTimerEnabled}"
                        itemLabel="#{msg.facs_timer}" >
                        <p:ajax process="@this"/>
                        </p:selectBooleanCheckbox>
                    <div id="continue-div">
                        <p:commandButton
value="#{msg.facs_continueToAnalysis}"
                            id="continue-button"

    disabled="#{feedbackAnalysisCategorySelectionManagedBean.isContinueDisabl
ed()}"

    action="#{feedbackAnalysisCategorySelectionManagedBean.checkCategories}"
                            process="@this category-sets" update="@form" />
                    </div>
                </h:form>
            </div>
            <h:outputScript>
                function focusCategory(category_set_class) {
                    var category_set = $(category_set_class);
                    if (category_set.length) {
                        setTimeout(function () {

    category_set.find(".category-text").last().focus();
                        }, 200);
                    }
                }

                function captureEnter(event) {
                    if (event.keyCode !== 13)
                        return true;
                    var focused = $(":focus");
                    if (focused.hasClass("category-text")) {
                        focused.blur();
                        var button = focused.parent().next("button");
                        if (button.length) button.click();
                    } else if (focused.hasClass("category-set-list-input")) {
                        var button = focused.next("button");
                        if (button.length) button.click();
                    }
                    return false;
                }
            </h:outputScript>
        </ui:define>
    </ui:composition>
</h:body>
</html>
```

# src/main/webapp/app/analysisrecordtable/index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:p="http://primefaces.org/ui"
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:pe="http://primefaces.org/ui/extensions"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">
<h:body>
    <ui:composition template="/WEB-INF/template.xhtml">
        <ui:define name="pageTitle" />
        <ui:define name="mainpage">
            <h:outputScript library="primefaces" name="jquery/jquery.js" />
            <h:outputScript library="js" name="locales.js" />
            <h:outputScript library="js" name="html2canvas.min.js" />
            <h:outputStylesheet library="css" name="analysisrecordtable.css" />
            <h:outputScript library="js" name="analysisrecordtable.js" />
            <h:outputScript library="js" name="table2csv.js" />
            <h1>#{msg.repo_title}</h1>
            <div id="container">
                <div id="content">
                    <ui:debug hotkey="x" />
                </div>
                <div id="continueToSummary">
                    <p:commandButton value="#{msg.repo_continue}"
                        action="#{feedbackAnalysisManagedBean.toSummary()}"

    rendered="#{!feedbackAnalysisManagedBean.containsUnclassifiedRecords()}">
                    </p:commandButton>
                    <p:commandButton value="#{msg.repo_continue}"
                        action="#{feedbackAnalysisManagedBean.toSummary()}"

    rendered="#{feedbackAnalysisManagedBean.containsUnclassifiedRecords()}">
                        <p:confirm header="#{msg.repo_question}"
                            message="#{msg.repo_warning}" icon="ui-icon-alert"
/>
                    </p:commandButton>
                </div>
                <div id="dataTableImage" >
                    <p:dataTable id="entries"

    value="#{feedbackAnalysisManagedBean.feedbackAnalysisEntity.records}"
                        var="record" sortBy="#{record.orderNumber}"
                        rowStyleClass="datatable" resizableColumns="true"
                        liveResize="false" styleClass="entries_dt"
reflow="true" tableStyle="width:auto">
                        <p:column headerText="#{msg.repo_nro}" >
                            <h:outputText class="tContent"
value="#{record.orderNumber}" />
                        </p:column>
                        <p:column headerText="#{msg.repo_feedback}"
id="givenFeedback">
                            <h:outputText class="tContent"
value="#{record.comment}" />
                        </p:column>
```

```
                        <c:forEach

    items="#{feedbackAnalysisManagedBean.feedbackAnalysisCategorySetsInUse.to
Array()}"
                            var="categorySet">
                            <p:column id="categories"
headerText="#{categorySet.label}">
                                <h:outputText class="tContent"

    value="#{analysisRecordTable.getSelectedCategorysName(record.getSelectedC
ategories(), categorySet)}" />
                            </p:column>
                        </c:forEach>
                        <p:column id="time"

    rendered="#{feedbackAnalysisManagedBean.isTimerEnabled}"
                            headerText="#{msg.repo_timestamp}">
                            <h:outputText

    value="#{feedbackAnalysisManagedBean.getLongAsTimeStamp(record.startTime)
}"
                                class="tContent" />
                        </p:column>
                        <p:column headerText="#{msg.repo_edit}"
exportable="false"
                            data-html2canvas-ignore="true" >
                            <span class="tContent"><p:commandButton
id="editButton"
                                    icon="fa fa-pencil"

    action="#{analysisRecordTable.edit(record.orderNumber)}" /></span>
                        </p:column>
                        <p:column exportable="false"
headerText="#{msg.repo_delete}"
                            data-html2canvas-ignore="true" >
                            <span class="tContent"><p:commandButton

    action="#{feedbackAnalysisManagedBean.delete(record.orderNumber)}"
                                    id="deleteButton" update="entries
reportButtons"
                                    icon="fa fa-trash"
oncomplete="sendImageAndCSV()">
                                    <p:confirm
header="#{msg.repo_confirmheader}"

    message="#{analysisRecordTable.getConfirm(record.orderNumber)}" />
                                </p:commandButton></span>
                        </p:column>
                    </p:dataTable>
                </div>

                <p:confirmDialog global="true" showEffect="fade"
hideEffect="fade">
                    <p:commandButton value="Yes" type="button"
                        styleClass="ui-confirmdialog-yes" icon="pi pi-check" />
                    <p:commandButton value="No" type="button"
```

```
                        styleClass="ui-confirmdialog-no" icon="pi pi-times" />
                </p:confirmDialog>
                <h:panelGroup id="reportButtons">
                    <h:form id="buttons">
                        <div id="downloadImage">
                            <p:commandButton value="#{msg.downloadImage}"
ajax="false"

    action="#{analysisRecordTable.downloadImage()}"></p:commandButton>
                        </div>
                    </h:form>
                    <div id="continueToSummary">
                        <p:commandButton value="#{msg.repo_continue}"

    action="#{feedbackAnalysisManagedBean.toSummary()}"
                            onclick="sendImageAndCSV()"

    rendered="#{!feedbackAnalysisManagedBean.containsUnclassifiedRecords()}">
                        </p:commandButton>
                        <p:commandButton value="#{msg.repo_continue}"

    action="#{feedbackAnalysisManagedBean.toSummary()}"

    rendered="#{feedbackAnalysisManagedBean.containsUnclassifiedRecords()}">
                            <p:confirm header="#{msg.repo_question}"
                                message="#{msg.repo_warning}"
icon="ui-icon-alert" />
                        </p:commandButton>
                    </div>
                </h:panelGroup>
            </div>
            <p:growl id="growl" widgetVar="growlWdgt" severity="info"
                showDetail="true" sticky="true" escape="false" />
            <p:confirmDialog global="true">
                <p:commandButton styleClass="ui-confirmdialog-yes"
                    value="#{msg.repo_yes}" type="button" />
                <p:commandButton styleClass="ui-confirmdialog-no"
                    value="#{msg.repo_no}" type="button" />
            </p:confirmDialog>
        </ui:define>
    </ui:composition>
</h:body>
</html>
```

``````

src/main/webapp/app/analyzer/index.xhtml

``````
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui"
```
```

```
    xmlns:f="http://xmlns.jcp.org/jsf/core">
<h:body>
    <ui:composition template="/WEB-INF/template.xhtml">
        <ui:define name="pageTitle">#{msg.ana_title}</ui:define>
        <ui:define name="mainpage">
            <h:outputScript library="primefaces" name="jquery/jquery.js" />
            <h:outputStylesheet library="css" name="analyzer.css" />
            <h:form id="timerControls"
                rendered="#{feedbackAnalysisManagedBean.isTimerEnabled}">
                <div id="timer_div">
                    <span id="timer_span"> <h:outputText id="txt_count"

    value="#{feedbackAnalysisManagedBean.getDurationAsString()}" />
                    </span> <span id="pause_span"> <p:selectBooleanButton

    value="#{feedbackAnalysisManagedBean.isTimerStopped}"
                            onLabel="#{msg.ana_continue_timer}" onIcon="fa
fa-play"
                            offLabel="#{msg.ana_pause_timer}" offIcon="fa
fa-pause" />
                    </span>
                    <p:poll id="timer" interval="1"
                        listener="#{feedbackAnalysisManagedBean.increment}"
                        update="txt_count" />
                </div>
            </h:form>
            <h:form id="form">
                <div id="content">
                    <div id="index">
                        <span class="arrows" id="toFirst"> <p:commandButton
                                value=" " id="firstAnalyze" icon="fa fa-fw
fa-fast-backward"

    action="#{feedbackAnalysisManagedBean.setCurrentRecord(1)}"

    disabled="#{feedbackAnalysisManagedBean.isNavigationDisabled(true)}"
                                process="@this :form:currentRecordComment
:form:categories"
                                update=":form" />
                        </span> <span class="arrows" id="toPrevious">
<p:commandButton
                                value=" " id="previousAnalyze"

    action="#{feedbackAnalysisManagedBean.setCurrentRecord(feedbackAnalysisMa
nagedBean.currentRecordNumber-1)}"

    disabled="#{feedbackAnalysisManagedBean.isNavigationDisabled(true)}"
                                process="@this :form:currentRecordComment
:form:categories"
                                update=":form" icon="fa fa-fw fa-arrow-left" />
                        </span>
                        <div id="pageNumber">
                            <h:outputText

    value="#{feedbackAnalysisManagedBean.currentRecordNumber}/#{feedbackAnaly
sisManagedBean.feedbackAnalysisEntity.records.size()}" />
```

```
                        </div>
                        <span class="arrows" id="toNext"> <p:commandButton
                                value=" " id="nextAnalyze" icon="fa fa-fw
fa-arrow-right"

    action="#{feedbackAnalysisManagedBean.setCurrentRecord(feedbackAnalysisMa
nagedBean.currentRecordNumber+1)}"

    disabled="#{feedbackAnalysisManagedBean.isNavigationDisabled(false)}"
                                process="@this :form:currentRecordComment
:form:categories"
                                update=":form" />
                        </span> <span class="arrows" id="toLast">
<p:commandButton
                                value=" " id="lastAnalyze" icon="fa fa-fw
fa-fast-forward"

    action="#{feedbackAnalysisManagedBean.setCurrentRecord(feedbackAnalysisMa
nagedBean.feedbackAnalysisEntity.records.size())}"

    disabled="#{feedbackAnalysisManagedBean.isNavigationDisabled(false)}"
                                process="@this :form:currentRecordComment
:form:categories"
                                update=":form" />
                        </span>
                    </div>
                    <h:panelGroup id="categories">
                        <ui:repeat

    value="#{feedbackAnalysisManagedBean.feedbackAnalysisCategorySetsInUse.to
Array()}"
                            var="categorySet">
                            <div id="categoryContent">
                                <div id="texts">
                                    <h:outputText value="#{categorySet.label}"
/>
                                </div>
                                <div id="buttons">
                                    <ui:repeat

    value="#{categorySet.categoryEntitys.values().toArray()}"
                                        var="category">
                                        <p:selectBooleanButton
value="#{category.inRecord}"
                                            onLabel="#{category.label.text}"

    offLabel="#{category.label.text}">
                                            <p:ajax

    listener="#{feedbackAnalysisManagedBean.setTimeStamp()}"
                                                update=":form:timestamp
:form:categories" />
                                        </p:selectBooleanButton>
                                    </ui:repeat>
                                </div>
                            </div>
```

```
                        </ui:repeat>
                        <h:panelGroup

    rendered="#{feedbackAnalysisManagedBean.isTimerEnabled}">
                            <div id="timestamp_div">
                                <span id="timestamp_name"> <h:outputText
                                        id="txt_timestamp"
value="#{msg.ana_timestamp}" />

                                </span> <span id="timestamp_span">
<p:commandLink
                                        onclick="PF('dlgTimeSlider').show()"
update=":timeSlider">
                                        <h:outputText id="timestamp"

    value="#{feedbackAnalysisManagedBean.getLongAsTimeStamp(feedbackAnalysisM
anagedBean.currentRecord.startTime)}" />
                                    </p:commandLink>
                                </span>
                            </div>
                        </h:panelGroup>
                    </h:panelGroup>
                </div>
                <div id="basicContent">
                    <div id="textArea">
                        <p:message id="commentError"
for="currentRecordComment" />
                        <p:inputTextarea id="currentRecordComment"

    value="#{feedbackAnalysisManagedBean.currentRecord.comment}"
                            validator="#{validationBean.validateLongString}">
                            <p:watermark for="currentRecordComment"
                                value="#{msg.ana_comment}" />
                            <p:ajax event="blur" process="@this"
update="commentError @this" />
                            <p:ajax event="keyup"

    listener="#{feedbackAnalysisManagedBean.setTimeStamp()}"
                                update=":form:timestamp" />
                        </p:inputTextarea>
                    </div>
                    <div id="basicControls">
                        <p:commandButton id="new" value="#{msg.ana_new}"

    action="#{feedbackAnalysisManagedBean.addRecord()}"
                            update=":form" />
                        <p:commandButton id="empty" value="#{msg.ana_empty}"

    action="#{feedbackAnalysisManagedBean.resetCurrentRecord()}"

    onclick="document.getElementById('form:currentRecordComment').value=''"
                            immediate="true" update=":form" />
                        <p:commandButton id="remove" value="#{msg.dlg_remove}"

    action="#{feedbackAnalysisManagedBean.delete(feedbackAnalysisManagedBean.
currentRecord.orderNumber)}"
```

```
                            immediate="true" update=":form">
                            <p:confirm header="#{msg.repo_confirmheader}"

    message="#{analysisRecordTable.getConfirm(feedbackAnalysisManagedBean.cur
rentRecord.orderNumber)}" />
                        </p:commandButton>
                        <p:commandButton id="continue"
value="#{msg.ana_continue_report}"
                            update=":form"

    action="#{feedbackAnalysisManagedBean.toRecordTable()}" />
                    </div>

                </div>
                <h:inputHidden

    validator="#{feedbackAnalysisManagedBean.validateNonEmptyRecord}" />
                <p:growl id="growl" showDetail="true" />
                            <p:confirmDialog global="true" showEffect="fade"
hideEffect="fade">
                <p:commandButton value="Yes" type="button"
                    styleClass="ui-confirmdialog-yes"  />
                <p:commandButton value="No" type="button"
                    styleClass="ui-confirmdialog-no"  />
            </p:confirmDialog>
            </h:form>

            <p:dialog widgetVar="dlgTimeSlider" position="top"
resizable="false"
                responsive="true" modal="true" fitViewport="true"
                class="coloredDialog">
                <h:form id="timeSlider">
                    <h:outputText id="timeOutput"

    value="#{feedbackAnalysisManagedBean.getLongAsTimeStamp(feedbackAnalysisM
anagedBean.currentRecord.startTime)}" />
                    <h:inputHidden id="secondsInput"

    value="#{feedbackAnalysisManagedBean.currentRecord.startTime}">
                    </h:inputHidden>
                    <p:slider for="secondsInput"

    minValue="#{feedbackAnalysisManagedBean.getMinTimeStampForCurrentRecord()
}"

    maxValue="#{feedbackAnalysisManagedBean.getMaxTimeStampForCurrentRecord()
}">
                        <p:ajax event="slideEnd" update="timeOutput
:form:timestamp"
                            process="secondsInput" />
                    </p:slider>
                </h:form>
            </p:dialog>
        </ui:define>

    </ui:composition>
```

```
</h:body>
</html>
``````

src/main/webapp/app/feedbackanalysissummary/index.xhtml

``````
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:c="http://java.sun.com/jsp/jstl/core">


<h:body>
    <ui:composition template="/WEB-INF/template.xhtml">

        <ui:define name="pageTitle">#{msg.ana_title}</ui:define>

        <ui:define name="mainpage">
            <h:outputScript library="primefaces" name="jquery/jquery.js" />
            <h:outputScript library="js" name="feedbackanalysissummary.js" />
            <h:outputStylesheet library="css"
name="feedbackanalysissummary.css" />
            <h:outputScript library="js" name="html2canvas.min.js" />
            <h:outputScript>
                function chartExtenderHideTicks() {
                    this.cfg.grid = {
                        background: 'transparent',
                        drawGridlines: false,
                        drawBorder: false,
                        shadow: false
                    };
                    this.cfg.axes.xaxis.showTicks = false;
                    this.cfg.axes.yaxis.showTicks = false;
                    this.cfg.title.fontSize = 1;
                    this.cfg.axes.yaxis.rendererOptions = {drawBaseline:
false};
                    this.cfg.axes.xaxis.rendererOptions = {drawBaseline: true};
                }
                function chartExtender() {
                    this.cfg.axes.xaxis.showTicks = false;
                    this.cfg.axes.xaxis.rendererOptions = {drawBaseline: true};
                    this.cfg.title.fontSize = 1;
                    this.cfg.grid = {
                        background: 'transparent',
                        drawGridlines: false,
                        drawBorder: false,
                        shadow: false
                    };
                }
            </h:outputScript>
```

```
            <f:event type="preRenderView"

    listener="#{feedbackAnalysisSummaryManagedBean.showObservationSavedMessag
e()}" />
            <h1>#{msg.ana_title} :

    #{feedbackAnalysisSummaryManagedBean.feedbackAnalysis.analysisName}</h1>
            <div id="container">
                <div class="hiddenButtons">
                    <h:form>
                        <div class="back">
                            <p:commandButton value="#{msg.fbsum_back}"
                                action="backtorecordtable"  />
                        </div>
                        <div class="save">
                            <p:commandButton value="#{msg.dlg_saveAndSend}"
                                rendered="#{sessionBean.identifiedUser}"
                                styleClass="save-button"
onclick="PF('dlgSave').show();"
                                process="@this">
                                <p:ajax update=":saveForm" resetValues="true"
/>
                            </p:commandButton>

                            <p:commandButton
value="#{msg.dlg_saveDialogHeaderAnalyzation}"
                                rendered="#{!sessionBean.identifiedUser}"
                                styleClass="save-button"
onclick="PF('dlgSave').show();"
                                process="@this">
                                <p:ajax update=":saveForm" resetValues="true"
/>
                            </p:commandButton>
                        </div>
                        <div class="end">
                            <p:commandButton value="#{msg.fbsum_end}"
action="tofrontpage">
                                <p:confirm header="#{msg.repo_confirmheader}"
                                    message="#{msg.fbsum_endconfirm}" />
                            </p:commandButton>
                        </div>
                    </h:form>
                </div>

                <h:form>
                    <div id="tableImage">
                        <table class="summaryTable">
                            <ui:repeat

    value="#{feedbackAnalysisSummaryManagedBean.tableInformations}"
                                var="tableInformation">
                                <tr>
                                    <td>
                                        <table class="summaryTable">
                                            <tr>
                                                <th
```

```
class="tableheader">#{tableInformation.feedbackAnalysisCategorySet}</th>
                                                    <th
class="tableheader">#{msg.fbsum_count}</th>
                                                </tr>
                                                <ui:repeat var="cat"
value="#{tableInformation.categories}"
                                                    varStatus="stat">
                                                    <tr>
                                                        <td style="width: 30%"
class="cat">#{cat}</td>
                                                        <td
class="count">#{tableInformation.counts.get(stat.index)}

    (#{feedbackAnalysisSummaryManagedBean.getPercentageAsString(tableInformat
ion.counts.get(stat.index))}
                                                                %)</td>
                                                    </tr>
                                                </ui:repeat>
                                            </table>
                                        </td>
                                    </tr>
                                </ui:repeat>

                            </table>
                        </div>
                        <p:commandButton value="#{msg.downloadImage}" ajax="false"

    action="#{feedbackAnalysisSummaryManagedBean.downloadImage('table')}" />
                        <p:separator />
                    </h:form>


                    <h:form id="charts">

                        <div id="charts_div">
                            <div id="chartsHeader">
                                <h2>#{msg.fbsum_charts}</h2>
                            </div>
                            <span id="bar"><p:selectBooleanCheckbox

    value="#{feedbackAnalysisSummaryManagedBean.renderBarChart}"
                                    id="barChart">#{msg.fbsum_barchart}
                <p:ajax update="charts" />
                                    </p:selectBooleanCheckbox></span> <br /> <span
id="pie"> <p:selectBooleanCheckbox

    value="#{feedbackAnalysisSummaryManagedBean.renderPieChart}"
                                    id="pieChart">#{msg.fbsum_piechart}
                <p:ajax update="charts" />
                                    </p:selectBooleanCheckbox></span>
                        </div>
                        <h:panelGroup

    rendered="#{feedbackAnalysisSummaryManagedBean.renderBarChart}"
                            layout="block" id="barModels">
                            <div id="bar_div">
```

```
                            <div id="barchartimage" style="">
                                <ui:repeat

    value="#{feedbackAnalysisSummaryManagedBean.barModels}"
                                    var="barModel" varStatus="myVarStatus">
                                    <span id="barchart"> <p:chart class="chart"
type="bar"
                                            model="#{barModel}"
                                            style="margin-bottom:
#{1.5*feedbackAnalysisSummaryManagedBean.countMaxCategories()}em"

    widgetVar="barchart#{myVarStatus.index}" />
                                    </span>
                                </ui:repeat>
                            </div>
                        </div>
                        <p>
                            <p:commandButton value="#{msg.downloadImage}"
ajax="false"

    action="#{feedbackAnalysisSummaryManagedBean.downloadImage('bar')}" />
                            <p:separator />
                        </p>
                    </h:panelGroup>

                    <div id="pie_div">
                        <h:panelGroup

    rendered="#{feedbackAnalysisSummaryManagedBean.renderPieChart}"
                            id="pieModels">
                            <div id="piechartimage">
                                <ui:repeat

    value="#{feedbackAnalysisSummaryManagedBean.pieModels}"
                                    var="pieModel" varStatus="myVarStatus">
                                    <span id="piechart"> <p:chart class="chart"
type="pie"
                                            model="#{pieModel}"
                                            style="width:300px; height:300px;
margin-bottom:
#{1.5*feedbackAnalysisSummaryManagedBean.countMaxCategories()}em"

    widgetVar="piechart#{myVarStatus.index}" />
                                    </span>
                                </ui:repeat>
                            </div>
                            <p:commandButton value="#{msg.downloadImage}"
ajax="false"

    action="#{feedbackAnalysisSummaryManagedBean.downloadImage('pie')}" />
                            <p:separator />
                        </h:panelGroup>
                    </div>
                </h:form>
```

```
                <div class="hiddenButtons">
                    <h:form>
                        <div class="back">
                            <p:commandButton value="#{msg.fbsum_back}"
                                action="backtorecordtable"/>
                        </div>
                        <div class="save">
                            <p:commandButton value="#{msg.dlg_saveAndSend}"
                                rendered="#{sessionBean.identifiedUser}"
                                styleClass="save-button"
onclick="PF('dlgSave').show();"
                                process="@this">
                                <p:ajax update=":saveForm" resetValues="true"
/>
                            </p:commandButton>

                            <p:commandButton
value="#{msg.dlg_saveDialogHeaderAnalyzation}"
                                rendered="#{!sessionBean.identifiedUser}"
                                styleClass="save-button"
onclick="PF('dlgSave').show();"
                                process="@this">
                                <p:ajax update=":saveForm" resetValues="true"
/>
                            </p:commandButton>
                        </div>
                        <div class="end">
                            <p:commandButton value="#{msg.fbsum_end}"
action="tofrontpage">
                                <p:confirm header="#{msg.repo_confirmheader}"
                                    message="#{msg.fbsum_endconfirm}" />
                            </p:commandButton>
                        </div>
                    </h:form>
                </div>

                <p:dialog header="#{msg.dlg_saveDialogHeaderAnalyzation}"
                    id="saveDialog" widgetVar="dlgSave" position="top"
                    resizable="false" responsive="true" fitViewport="true"
                    visible="#{facesContext.validationFailed}"
                    styleClass="coloredDialog">
                    <h:form id="saveForm" styleClass="dialog-form ui-fluid">
                        <p:selectManyCheckbox id="basic"
                            rendered="#{sessionBean.identifiedUser}"
columns="1"
                            required="true" requiredMessage="" layout="grid"

    value="#{feedbackAnalysisSummaryManagedBean.selectedSaveOperations}">
                            <p:ajax process="@this" update="@this input-email"
/>
                            <f:selectItem itemLabel="#{msg.dlg_saveOption}"
itemValue="save"
                                itemDisabled="#{!sessionBean.identifiedUser}"
/>
                            <f:selectItem
itemLabel="#{msg.dlg_downloadOption}"
```

```
                                itemValue="download"
itemDisabled="#{!sessionBean.loggedIn}" />
                            <f:selectItem itemLabel="#{msg.dlg_mailOption}"
itemValue="mail"
                                itemDisabled="#{!sessionBean.identifiedUser}"
/>
                        </p:selectManyCheckbox>
                        <p:selectManyCheckbox id="anonymityUserBoxes"
                            rendered="#{!sessionBean.identifiedUser}"
columns="1"
                            required="true" requiredMessage="" layout="grid"

    value="#{feedbackAnalysisSummaryManagedBean.selectedSaveOperations}">
                            <p:ajax process="@this" update="@this" />
                            <f:selectItem
itemLabel="#{msg.dlg_downloadOption}"
                                itemValue="download"
itemDisabled="#{!sessionBean.loggedIn}" />
                        </p:selectManyCheckbox>
                        <h:outputLabel for="input-name"
                            value="#{msg.asum_analyzationName}">
                            <h:outputText
rendered="#{sessionBean.identifiedUser}"
                                value=" (#{msg.dlg_defaultFileName})" />
                        </h:outputLabel>
                        <p:message id="nameError" for="input-name" />
                        <p:inputText id="input-name" required="true"
                            requiredMessage="#{msg.dlg_notEmpty}"

    value="#{feedbackAnalysisManagedBean.feedbackAnalysisEntity.analysisName}
"

    validator="#{validationBean.validateShortString}">

                            <f:ajax event="keyup" execute="@this"
render="nameError" />
                        </p:inputText>

                        <h:outputLabel for="input-description"
                            value="#{msg.asum_analyzationDescription}" />
                        <p:message id="descriptionError"
for="input-description" />
                        <p:inputTextarea id="input-description"

    value="#{feedbackAnalysisManagedBean.feedbackAnalysisEntity.description}"
                            validator="#{validationBean.validateLongString}"
rows="3"
                            cols="16" autoResize="true">
                            <f:ajax event="keyup" execute="@this"
render="descriptionError" />
                        </p:inputTextarea>

                        <h:outputLabel for="input-target"
                            value="#{msg.asum_analyzationTarget}" />
                        <p:message id="targetError" for="input-target" />
                        <p:inputText id="input-target"
```

```
    value="#{feedbackAnalysisManagedBean.feedbackAnalysisEntity.targetOfAnaly
sis}"

    validator="#{validationBean.validateShortString}">
                            <f:ajax event="keyup" execute="@this"
render="targetError" />
                        </p:inputText>


                        <p:outputPanel
rendered="#{sessionBean.identifiedUser}">
                            <h:outputLabel for="input-email"
value="#{msg.dlg_giveEmail}" />
                            <p:message id="emailError" for="input-email" />
                            <p:inputText id="input-email"

    required="#{feedbackAnalysisSummaryManagedBean.isSelected('mail')}"
                                requiredMessage="#{msg.dlg_notEmpty}"

    disabled="#{!feedbackAnalysisSummaryManagedBean.isSelected('mail')}"

    value="#{feedbackAnalysisSummaryManagedBean.emailAddress}">
                                <f:validator validatorId="emailValidator" />
                                <f:ajax event="blur" execute="@this"
render="emailError" />
                            </p:inputText>
                        </p:outputPanel>
                        <p:commandButton value="#{msg.dlg_save}"
icon="ui-icon-check"
                            ajax="false"
action="#{feedbackAnalysisSummaryManagedBean.save}"
                            validateClient="true">
                        </p:commandButton>
                        <p:commandButton value="#{msg.dlg_cancel}"
icon="ui-icon-close"
                            onclick="PF('dlgSave').hide();" />

                    </h:form>
                </p:dialog>

                <p:growl id="growl" widgetVar="growlWdgt" severity="info"
                    showDetail="true" sticky="false" escape="false" />

            </div>
            <p:confirmDialog global="true">
                <p:commandButton styleClass="ui-confirmdialog-yes"
                    value="#{msg.repo_yes}" type="button" />
                <p:commandButton styleClass="ui-confirmdialog-no"
                    value="#{msg.repo_no}" type="button" />
            </p:confirmDialog>
        </ui:define>

    </ui:composition>
</h:body>
</html>
```

```

# src/main/webapp/error/404.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:body>
    <ui:composition template="/WEB-INF/template.xhtml">
        <ui:define
name="pageTitle">#{errormessages.errorPageTitle}</ui:define>
        <ui:define name="mainpage">
            <link type="text/css" rel="stylesheet"

    href="/javax.faces.resource/primefaces.css.xhtml?ln=primefaces&v=5.3"
/>
            <h:outputText value="#{errormessages.notFoundMessage}" />
            <br />
            <h:outputLink value="/">
                <h:outputText value="#{errormessages.toFrontPage}" />
            </h:outputLink>
        </ui:define>
    </ui:composition>
</h:body>
</html>
```

# src/main/webapp/error/500.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets">


<h:body>
    <ui:composition template="/WEB-INF/template.xhtml">
        <ui:define
name="pageTitle">#{errormessages.errorPageTitle}</ui:define>
        <ui:define name="mainpage">
            <link type="text/css" rel="stylesheet"

    href="/javax.faces.resource/primefaces.css.xhtml?ln=primefaces&v=5.3"
/>
            <h:outputText value="#{errormessages.errorMessageToUser}" />
```
```

```
            <br />
            <h:outputLink value="/">
                <h:outputText value="#{errormessages.toFrontPage}" />
            </h:outputLink>
        </ui:define>
    </ui:composition>
</h:body>
</html>
```