Санкт-Петербургский государственный политехнический университет

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

ОТЧЕТ

о курсовом проекте

по дисциплине: «Программное обеспечение распределенных вычислительных систем»

Тема работы: «Информационная система научного журнала»

Работу выполнил студент

63501/3 *Алексюк А.О.*

Преподаватель

_____ *Стручков И.В.*

Санкт-Петербург
2016

1. Анализ задания

В рамках курса было необходимо разработать приложение для распределенных вычислительных систем. Приложение должно удовлетворять требованиям открытости, масштабируемости и прозрачности, применять технологии EJB и JPA и использовать Web-интерфейс для взаимодействия с пользователем.

Было решено разработать информационную систему для научного журнала.

1.1. Модель предметной области

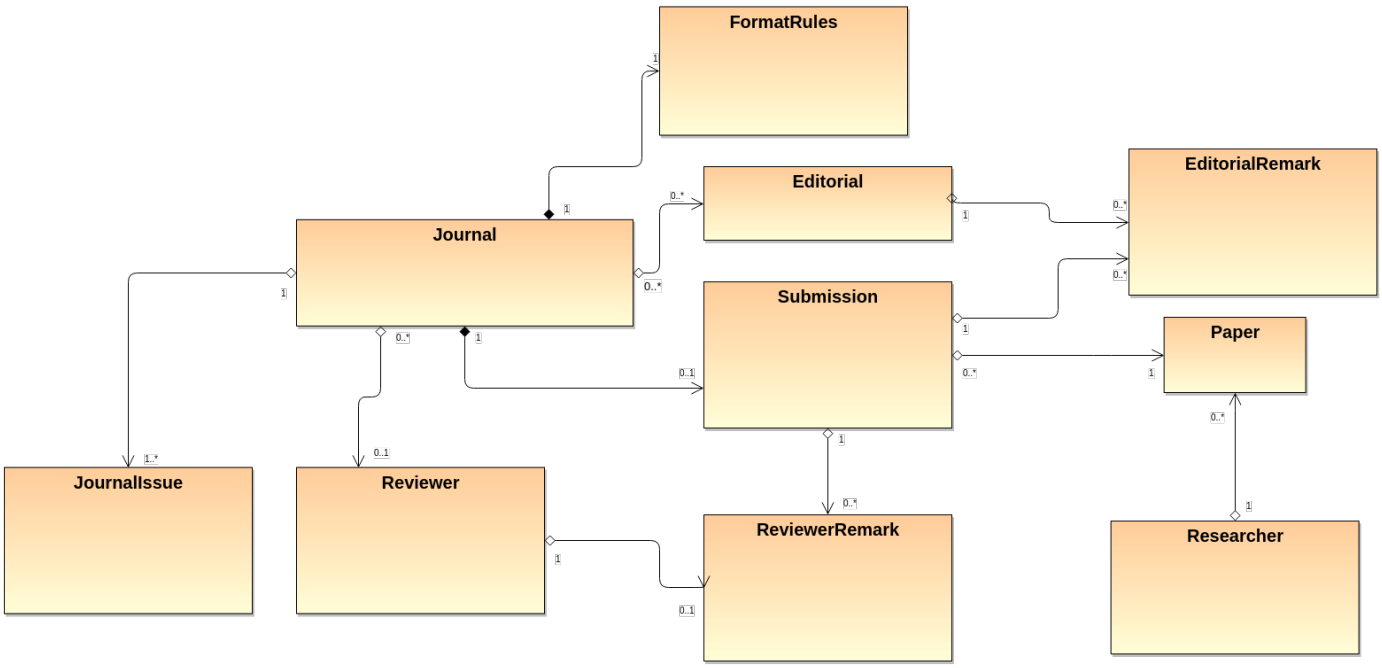

Рис. 1: Модель предметной области

1.2. Роли

В проекте выделено 3 роли: исследователь, редакция и рецензент:

- Исследователь
  - Разрабатывает научную тему
  - Пишет статью по ней
  - Принимает замечания по ней
  - *Цель:* Чтобы его статья была опубликована в журнале
- Рецензент
  - Выбирается редакцией
  - Получает статьи для просмотра
  - Дает оценку статье (стоит ли принимать для публикации)

- Высказывает замечания, возникшие при прочтении статьи
- *Цель:* Выбрать подходящие статьи

- Редакция
  - Принимает статьи
  - Устанавливает правила принятия статей
  - Подбирает рецензентов
  - Связывает рецензентов и авторов
  - Корректирует статьи, если необходимо
  - Издает журнал с помощью типографии
  - *Цель:* Принять качественные статьи, заработать на продаже журналов

2. Варианты использования

2.1. Написание и подача статьи

1) Исследователь запрашивает веб-страницу, предназначенную для добавления новой публикации.

2) Система запрашивает из БД список исследователей, и помещает их имена на веб-страницу в виде выпадающего списка, с возможностью выбора одного из них.
   Также система запрашивает из БД список всех публикаций, и для каждой публикации помещает на страницу её заголовок, реферат и содержимое

3) Исследователь на странице в выпадающем списке выбирает своё имя и нажимает кнопку «Отправить».

4) Система фиксирует у себя имя текущего исследователя и перерисовывает страницу, добавляя на неё имя исследователя.

5) Автор заполняет на странице поля «Заголовок», «Реферат» (Abstract) и «Содержимое статьи» (Content). После заполнения всех полей автор нажимает кнопку «Отправить».

6) Система создает объект типа «Статья» (Paper), заполняя его присланными пользователем данными. В качестве автора указывается текущий исследователь. После этого система создает объект типа «Подача» (Submission), указывая в качестве его атрибута только что созданную статью. Оба объекта помещаются в БД.

7) Система перерисовывает страницу, в списке статей появляется только что добавленная статья

2.2. Проверка статьи редактором журнала

1) Редактор запрашивает веб-страницу, предназначенную для добавления новой оценки.

2) Система запрашивает из БД список всех не просмотренных редакцией публикаций, и для каждой публикации помещает на страницу её заголовок, реферат и содержимое. На страницу рядом с каждой статьей в списке помещается набор кнопок для выставления решения.

3) Редактор просматривает статьи и решает, что одна из статей соответствует тематике журнала и удовлетворяет правилам оформления. Редактор нажимает кнопку «Одобрить» около статьи в списке.

   - Альтернатива: Редакция отказывает в приеме статьи по причине недоработок в статье (например, проблемах с форматированием). Редактор в текстовом поле указывает список замечаний, который будет передан автору, и нажимает кнопку «Отправить на доработку»
   - Альтернатива: Редакция отказывает в приеме по причине несоответствия тематике журнала. Редактор в текстовом поле указывает список замечаний, который будет передан автору, и нажимает кнопку «Статья для другого журнала».

4) Система создает объект типа «Оценка редактора» (EditorialReview), указывая в нем оценку и комментарий редактора. Система получает из БД объект «Подача» (Submission) для указанной статьи и добавляет в него объект «Оценка редактора». Также у объекта подачи устанавливается состояние, соответствующее выбору редактора (Одобрить/Доработать/Перенаправить).

5) Система перерисовывает страницу, из списка статей исчезает только что оцененная статья.

2.3. Рецензирование

1) Рецензент запрашивает веб-страницу, предназначенную для добавления новой рецензии.

2) Система запрашивает из БД список рецензентов, и помещает их имена на веб-страницу в виде выпадающего списка, с возможностью выбора одного из них.
   Также система запрашивает из БД список всех одобренных редакцией публикаций, и для каждой публикации помещает на страницу её заголовок, реферат и содержимое. На страницу рядом с каждой статьей в списке помещается набор кнопок для выставления решения.

3) Рецензент на странице в выпадающем списке выбирает своё имя и нажимает кнопку «Отправить».

4) Система фиксирует у себя имя текущего рецензента и перерисовывает страницу, добавляя на неё имя исследователя.

5) Рецензент просматривает статьи и решает, что одна из статей достойна публикации в журнале. В поле «Замечания» на странице сайта он указывает свою рецензию. Рецензент нажимает кнопку «Accept», статья отмечается в системе как принятая и исчезает из списка не просмотренных.

    - Альтернатива: Рецензент имеет замечания к статье, но допускает её для публикации (Neutral). В поле «Замечания» на странице сайта он указывает свою рецензию. Рецензент нажимает кнопку «Neutral».
    - Альтернатива: Рецензент имеет замечания к статье и не допускает её для публикации (Reject). В поле «Замечания» на странице сайта он указывает свою рецензию. Рецензент нажимает кнопку «Reject».

6) Система создает объект типа «Оценка рецензента» (ReviewerRemark), указывая в нем оценку и комментарий рецензента. Система получает из БД объект «Подача» (Submission) для указанной статьи и в качестве атрибута «Рецензия» указывает у него только что созданный объект. Также у объекта подачи устанавливается состояние, соответствующее выбору рецензента (Accept, Neutral, Reject).

7) Система перерисовывает страницу, из списка статей исчезает только что оцененная статья.

2.4. Диаграмма вариантов использования

Рис. 2: Диаграмма вариантов использования

2.5. Диаграмма последовательностей

2.5.1. Получение списка статей

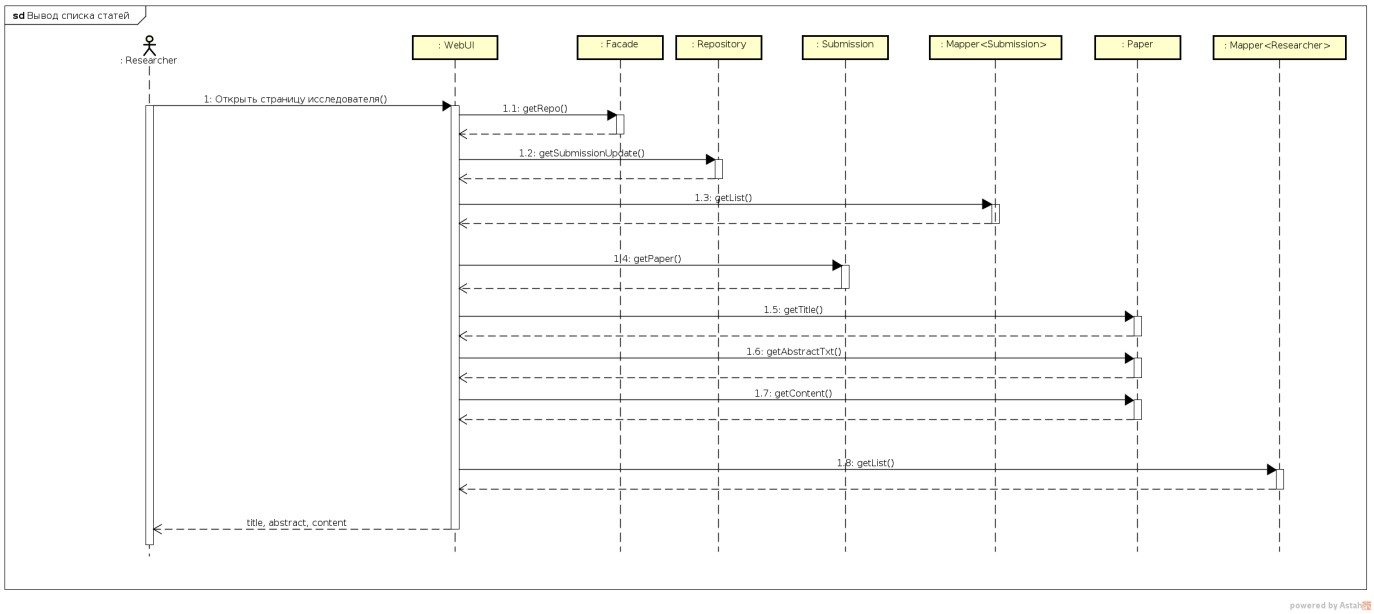

Рис. 3

2.5.2. Подача статей

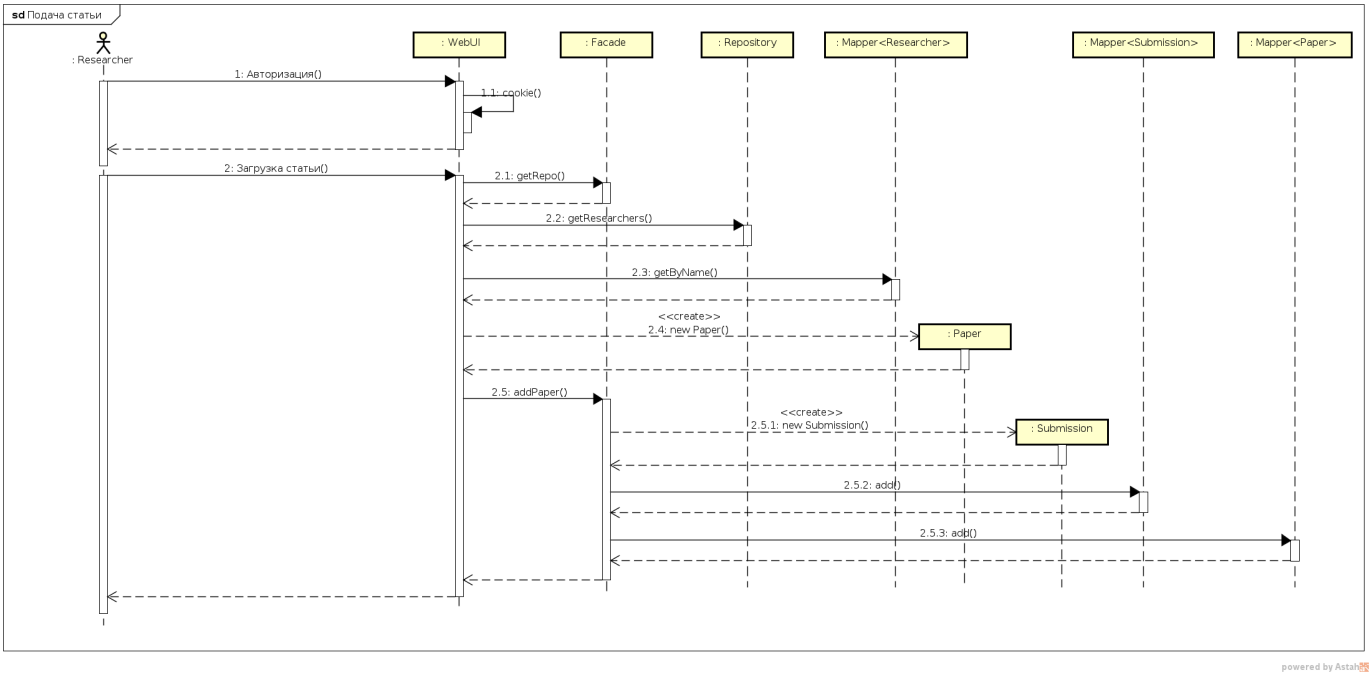

Рис. 4

2.5.3. Редакция

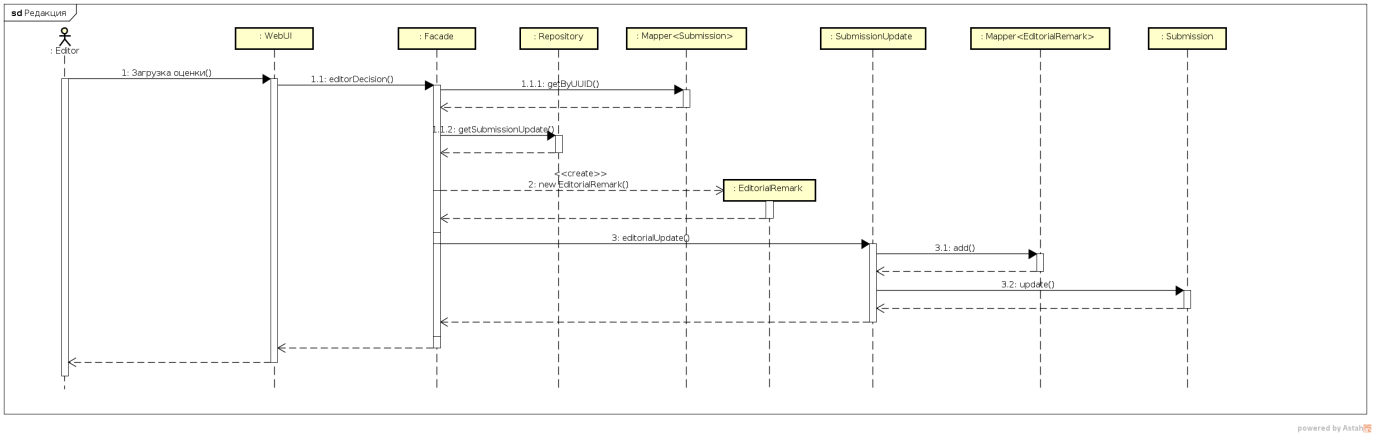

Рис. 5

2.5.4. Рецензирование

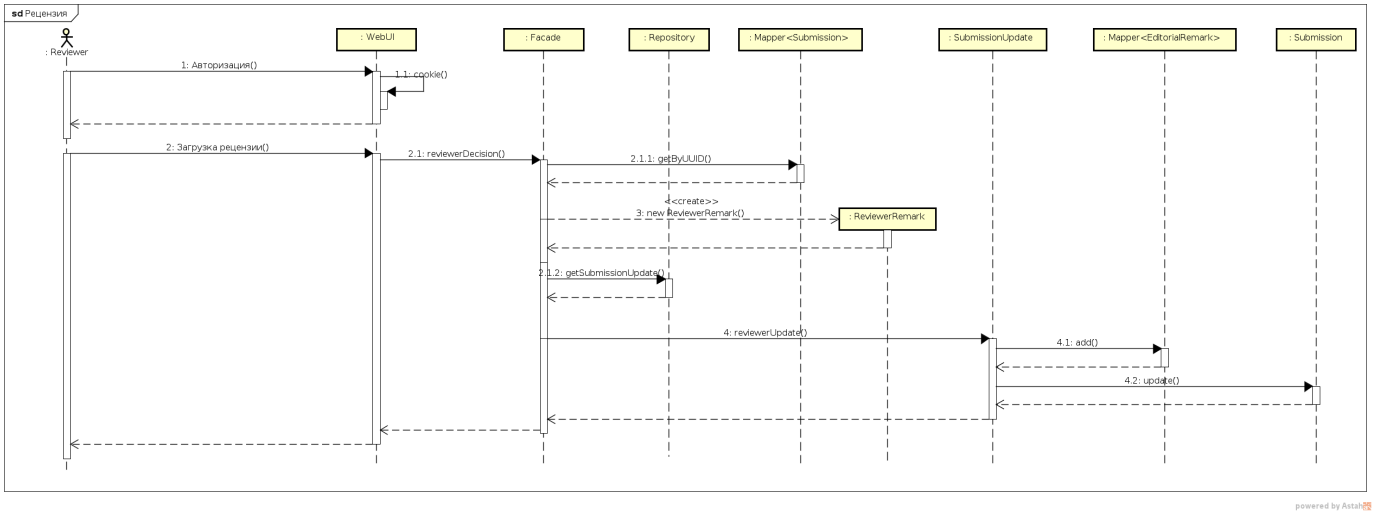

Рис. 6

3. Реализация задания с помощью технологии EJB

3.1. Объектно-ориентированное проектирование с учётом особенностей технологии

Ниже приведена диаграмма классов для пакета objects, в котором содержатся классы, соответствующие сущностям предметной области. Альтернативы из вариантов использования представлены в приложении в виде перечислений Decision и Mark.

Для отслеживания состояния статьи используется перечисление State, имеющее варианты для каждого этапа обработки статьи.

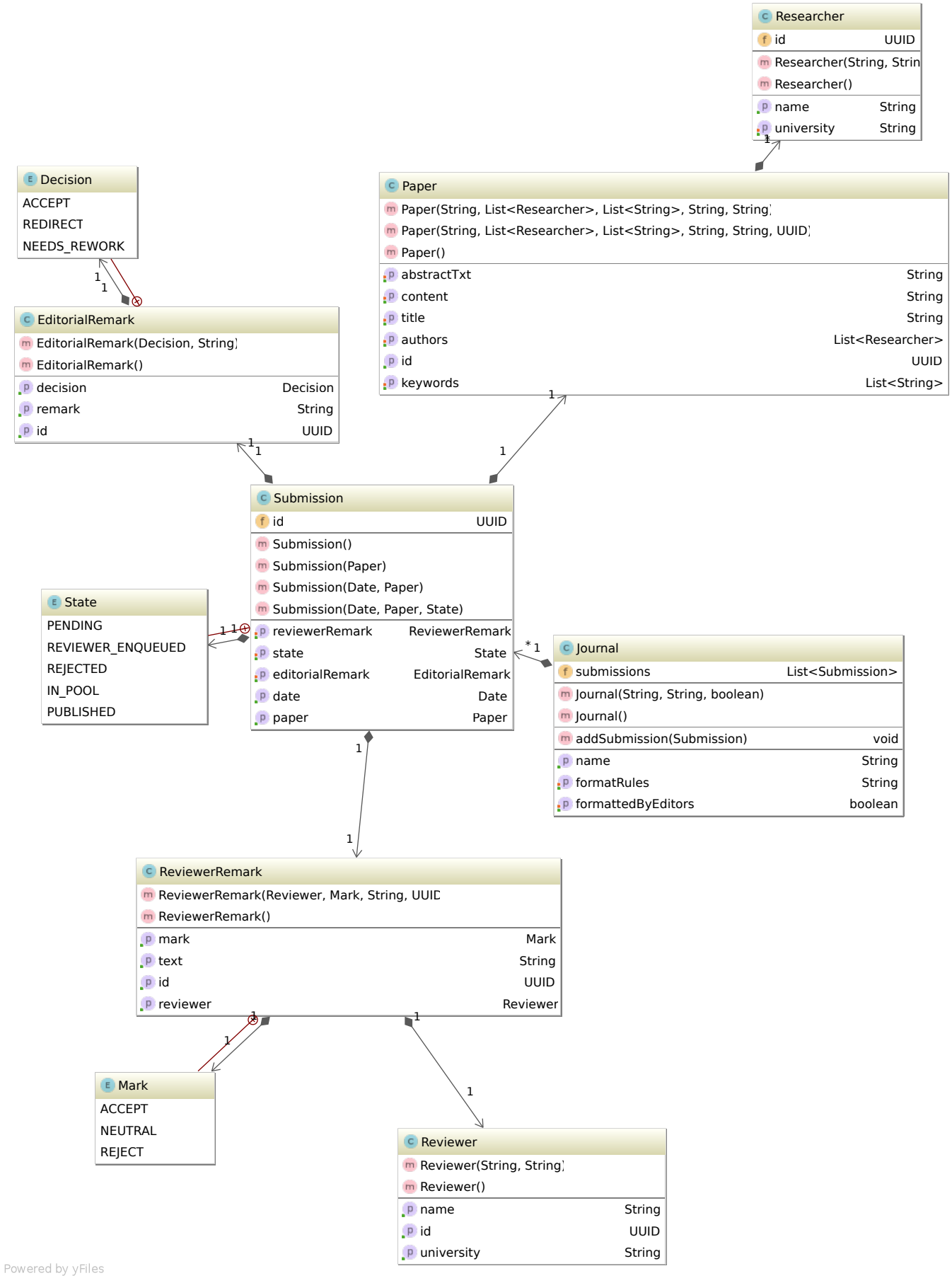

Рис. 7: Диаграмма класса для пакета objects

Ниже приведена диаграмма классов для пакетов services и repository, содержащих различные сервисы, используемые в приложении.

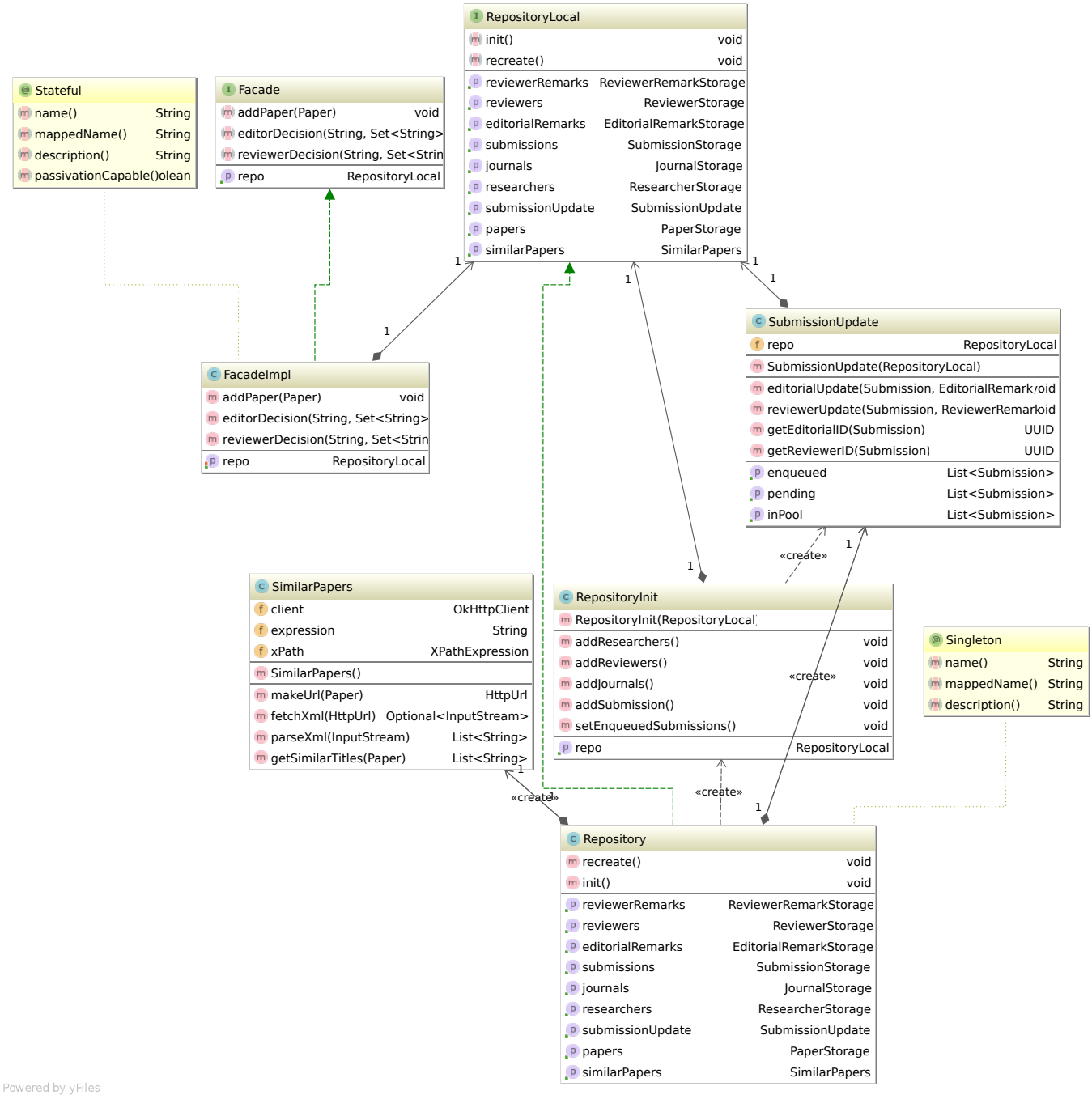

Рис. 8

На рисунке 9 показана диаграмма классов для слоя хранения.

Storage
add(T) T
recreate() void
list List<T>
ResearcherStorage
get(String) Optional<Researcher>
ReviewerRemarkStorage
get(UUID) Optional<ReviewerRemark>
ReviewerStorage
get(String) Optional<Reviewer>
get(UUID) Optional<Reviewer>
Mapper
repo Repository
em EntityManager
Mapper(Repository) T
add(T) void
recreate() Class<T>
getEntityClass() List<T>
list
SubmissionStorage
update(Submission) void
get(State) List<Submission>
get(UUID) Optional<Submission>
PaperStorage
get(UUID) Optional<Paper>
EditorialRemarkStorage
get(UUID) Optional<EditorialRemark>
JournalStorage
formattedByEditors Optional<Journal>
ResearcherMapper
ResearcherMapper(Repository)
getEntityClass() Class<Researcher>
ReviewerRemarkMapper
ReviewerRemarkMapper(Repository
getEntityClass() Class<ReviewerRemark>
ReviewerMapper
ReviewerMapper(Repository)
getEntityClass() Class<Reviewer>
get(String) Optional<Reviewer>
SubmissionMapper
SubmissionMapper(Repository)
getEntityClass() Class<Submission>
update(Submission) void
PaperMapper
PaperMapper(Repository)
getEntityClass() Class<Paper>
get(UUID) Optional<Paper>
EditorialRemarkMapper
EditorialRemarkMapper(Repository
getEntityClass() Class<EditorialRemark>
Powered by yFiles

Рис. 9

3.2. Описание программы

Было решено, что в приложении будет два bean-а:

1) Facade - класс для реализации паттерна проектирования «Фасад». Соответствующий Bean было решено сделать @Stateful, чтобы исключить проблемы, связанные с одновременным использованием одного экземпляра класса несколькими клиентами.

2) Repository - класс-репозиторий, содержащий ссылки на объекты-Mapper-ы. Так как все клиенты используют общий набор Mapper-ов, класс Repository был помечен аннотацией @Singleton.

Класс Facade содержит следующие методы:

- getRepo() - возвращает объект-репозиторий. Внутри класса Facade содержится поле repo, которое автоматически заполняется EJB-контейнером при создании нового экземпляра класса. Для этого поле repo было помечено аннотацией @EJB.

- addPaper() - добавляет статью в базу данных, и создает для неё новый объект «Подача» (Submission).

- editorDecision(String uuidString, Set<String> params, String remarkText) - для указанной статьи устанавливает решение редактора и добавляет примечание

- reviewerDecision(String uuidString, Set<String> params, String user, String remarkTe - для указанной статьи устанавливает решение рецензента и добавляет примечание

Для реализации Web-интерфейса была использована библиотека Spark, которая в свою очередь основана на API Servlet. Логика веб-интерфейса реализована в классе WebUI. В этом классе имеется только один метод, init, в котором устанавливаются обработчики для различных адресов. Для настройки сервера приложений использовался файл web.xml.

ORM реализован с помощью фреймворка Hibernate. Имеется класс Mapper, который осуществляет взаимодействие с EntityManager-ом. Пример получения данных из БД:

```
1 CriteriaQuery<T> query = em.getCriteriaBuilder().createQuery(
     getEntityClass());
2 Root<T> root = query.from(getEntityClass());
3 query.select(root);
4 return em.createQuery(query).getResultList();
```

4. Тестирование

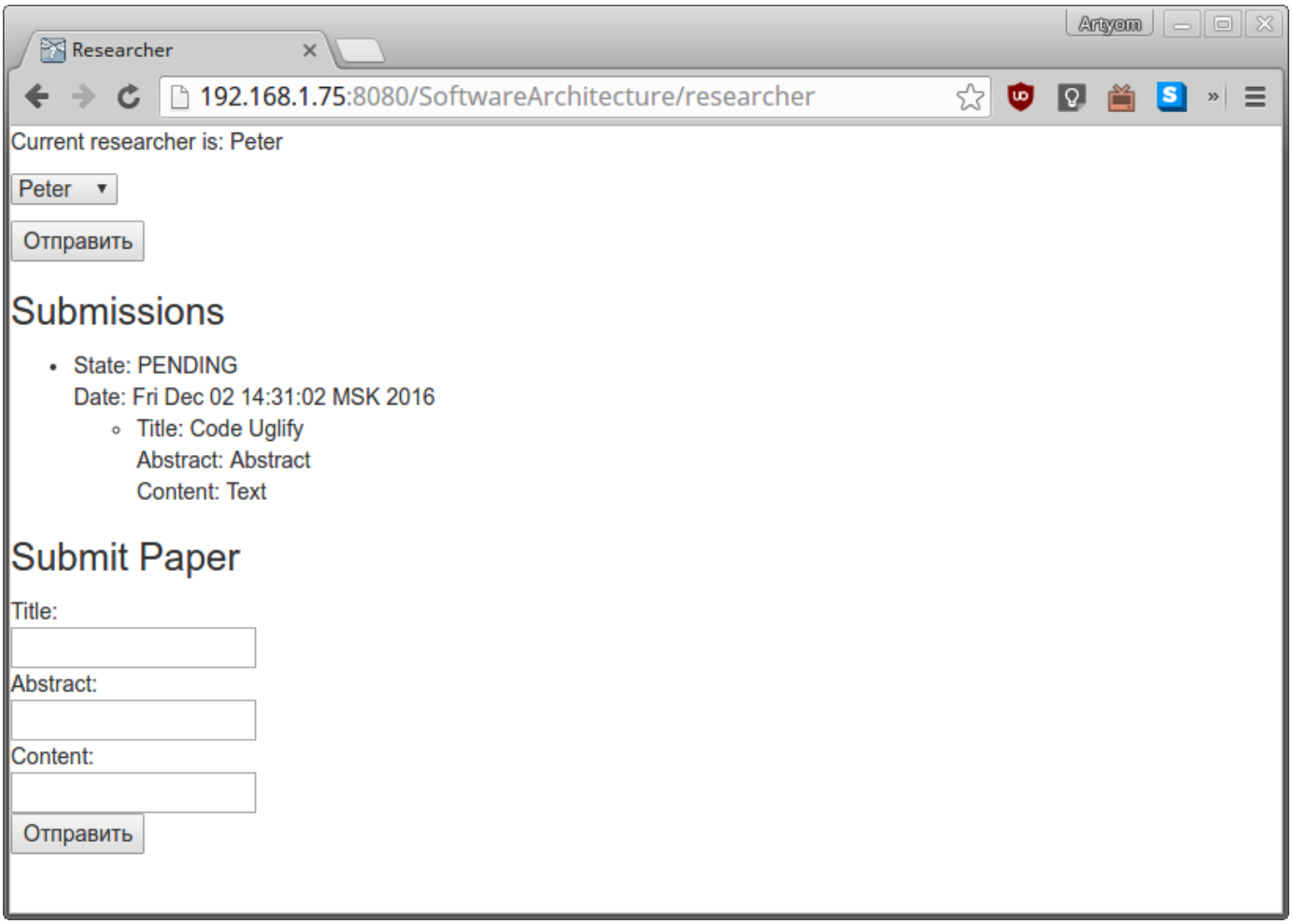

Рис. 10: Страница исследователя

Тестирование страницы исследователя:

Вариант использования	**Ожидаемый результат**	**Фактический результат**
Выбор исследователя из списка и нажатие кнопки отправить	Выбранный пользователь становится текущим	Над полем выбора пользователя появляется строчка: «Current researcher is: Имя исследователя»
Заполнение полей Title, Abstract и Content и нажатие кнопки «Отправить»	В списке статей появится новая статья	В разделе Submissions появляется новая запись

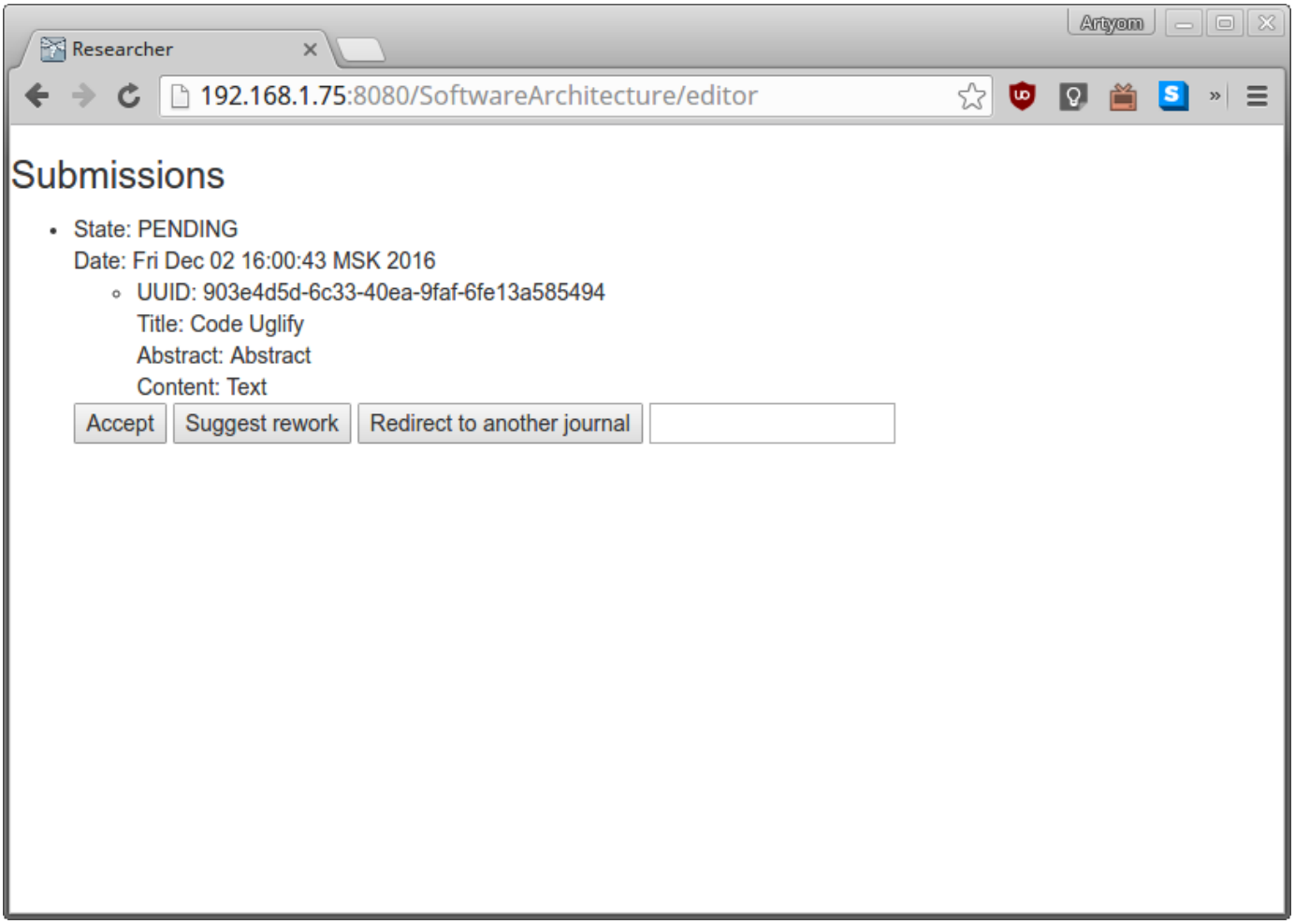

Рис. 11: Страница редактора

Тестирование страницы редактора:

Вариант использования	**Ожидаемый результат**	**Фактический результат**
Выбор статьи из списка и нажатие кнопки Accept	Статья перейдет в состояние «Принято»	Статья исчезает из списка на странице (помечается как просмотренная), на странице исследователя появляется отметка «Review remark: ACCEPT Note: Сообщение»
Выбор статьи из списка и нажатие кнопки Needs Rework	Статья перейдет в состояние «Требуется исправление»	Статья исчезает из списка на странице (помечается как просмотренная), на странице исследователя появляется отметка «Review remark: NEEDS_REWORK Note: Сообщение»

Выбор статьи из списка и нажатие кнопки Redirect to another journal	Статья перейдет в состояние «Статья для другого журнала»	Статья исчезает из списка на странице (помечается как просмотренная), на странице исследователя появляется отметка «Review remark: REDIRECT Note: Сообщение»

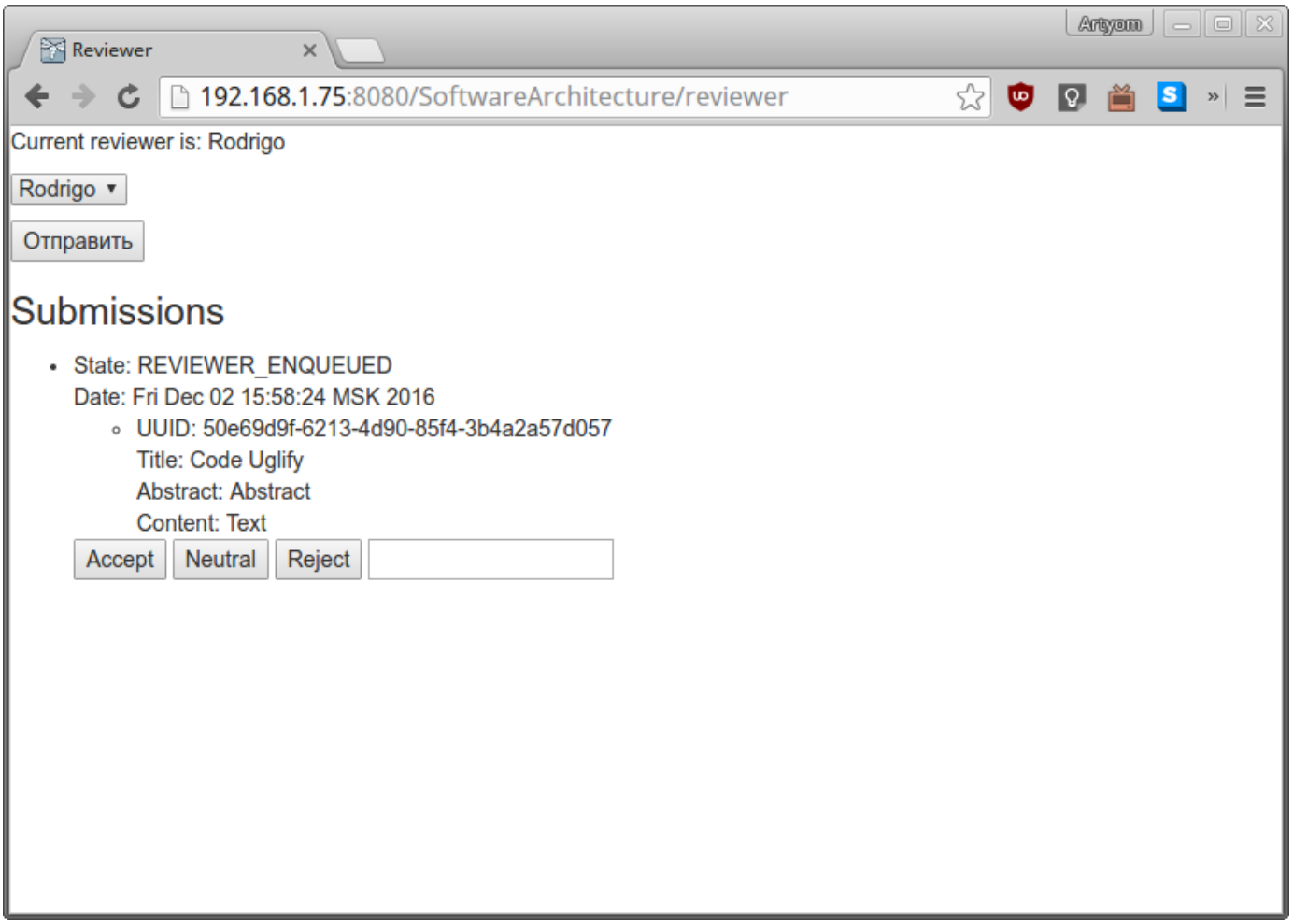

Рис. 12: Страница рецензента

Тестирование страницы рецензента:

Вариант использования	**Ожидаемый результат**	**Фактический результат**

Выбор статьи из списка и нажатие кнопки Accept	Статья перейдет в состояние «Готова для печати (In pool)» для последующей отправки в журнал	Статья исчезает из списка на странице (помечается как просмотренная), на странице исследователя появляется отметка «Review remark: ACCEPT Note: Сообщение», статья доступна для выгрузки в журнал
Выбор статьи из списка и нажатие кнопки Neutral	Статья перейдет в состояние «Готова для печати (In pool)» для последующей отправки в журнал	Статья исчезает из списка на странице (помечается как просмотренная), на странице исследователя появляется отметка «Review remark: NEUTRAL Note: Сообщение», статья доступна для выгрузки в журнал
Выбор статьи из списка и нажатие кнопки Reject	Статья перейдет в состояние «Отклонена (Rejected)»	Статья исчезает из списка на странице (помечается как просмотренная), на странице исследователя появляется отметка «Review remark: REJECTED Note: Сообщение»

4.1. Инструкция системного администратора

Исходный код веб-приложения доступен по адресу:
`https://github.com/h31/SoftwareArchitecture`
Для корректной работы проекта требуется установить следующее ПО:

- Пакет Java Runtime Environment 8
- Сервер приложений WildFly 10

Для сборки проекта необходимо выполнить команду ./gradlew war:

```
$ ./gradlew war
:clean
:compileJava
:processResources
:classes
:war
```

```
7
8  BUILD SUCCESSFUL
9
10 Total time: 0.946 secs
```

Собранный war-файл доступен по следующему пути: build/libs/SoftwareArchitecture.war

После сборки необходимо установить и настроить WildFly. С помощью скрипта add-user.sh добавим пользователя-администратора:

```
1  artyom@artyom-MSI:~/Tools/wildfly-10.1.0.Final$ bin/add-user.sh
2
3  What type of user do you wish to add?
4   a) Management User (mgmt-users.properties)
5   b) Application User (application-users.properties)
6  (a):
7
8  Enter the details of the new user to add.
9  Using realm 'ManagementRealm' as discovered from the existing
       property files.
10 Username : user
11 Password recommendations are listed below. To modify these
       restrictions edit the add-user.properties configuration file.
12  - The password should be different from the username
13  - The password should not be one of the following restricted values
        {root, admin, administrator}
14  - The password should contain at least 8 characters, 1 alphabetic
        character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
15 Password :
16 WFLYDM0099: Password should have at least 8 characters!
17 Are you sure you want to use the password entered yes/no? yes
18 Re-enter Password :
19 What groups do you want this user to belong to? (Please enter a comma
       separated list, or leave blank for none)[  ]:
20 About to add user 'user' for realm 'ManagementRealm'
21 Is this correct yes/no? yes
22 Added user 'user' to file '/home/artyom/Tools/wildfly-10.1.0.Final/
       standalone/configuration/mgmt-users.properties'
23 Added user 'user' to file '/home/artyom/Tools/wildfly-10.1.0.Final/
       domain/configuration/mgmt-users.properties'
24 Added user 'user' with groups  to file '/home/artyom/Tools/wildfly
       -10.1.0.Final/standalone/configuration/mgmt-groups.properties'
25 Added user 'user' with groups  to file '/home/artyom/Tools/wildfly
       -10.1.0.Final/domain/configuration/mgmt-groups.properties'
26 Is this new user going to be used for one AS process to connect to
       another AS process?
27 e.g. for a slave host controller connecting to the master or for a
       Remoting connection for server to server EJB calls.
28 yes/no? no
```

После добавления пользователя можно запустить сам сервер приложений с помощью скрипта standalone.sh

```
1  artyom@artyom-MSI:~/Tools/wildfly-10.1.0.Final$ bin/standalone.sh
```

```
2  ================================================================================
3
4    JBoss Bootstrap Environment
5
6    JBOSS_HOME: /home/artyom/Tools/wildfly-10.1.0.Final
7
8    JAVA: /usr/lib/jvm/java-8-oracle/bin/java
9
10   JAVA_OPTS:  -server -Xms64m -Xmx512m -XX:MetaspaceSize=96M -XX:
       MaxMetaspaceSize=256m -Djava.net.preferIPv4Stack=true -Djboss.
       modules.system.pkgs=org.jboss.byteman -Djava.awt.headless=true
11
12 ================================================================================

13 ...
14 13:13:05,239 INFO  [org.jboss.as] (Controller Boot Thread)
     WFLYSRV0060: Http management interface listening on http://
     127.0.0.1:9990/management
15 13:13:05,239 INFO  [org.jboss.as] (Controller Boot Thread)
     WFLYSRV0051: Admin console listening on http://127.0.0.1:9990
```

WildFly выведет на консоль адрес страницы для управления сервером приложения. Зайдем на неё с помощью браузера:

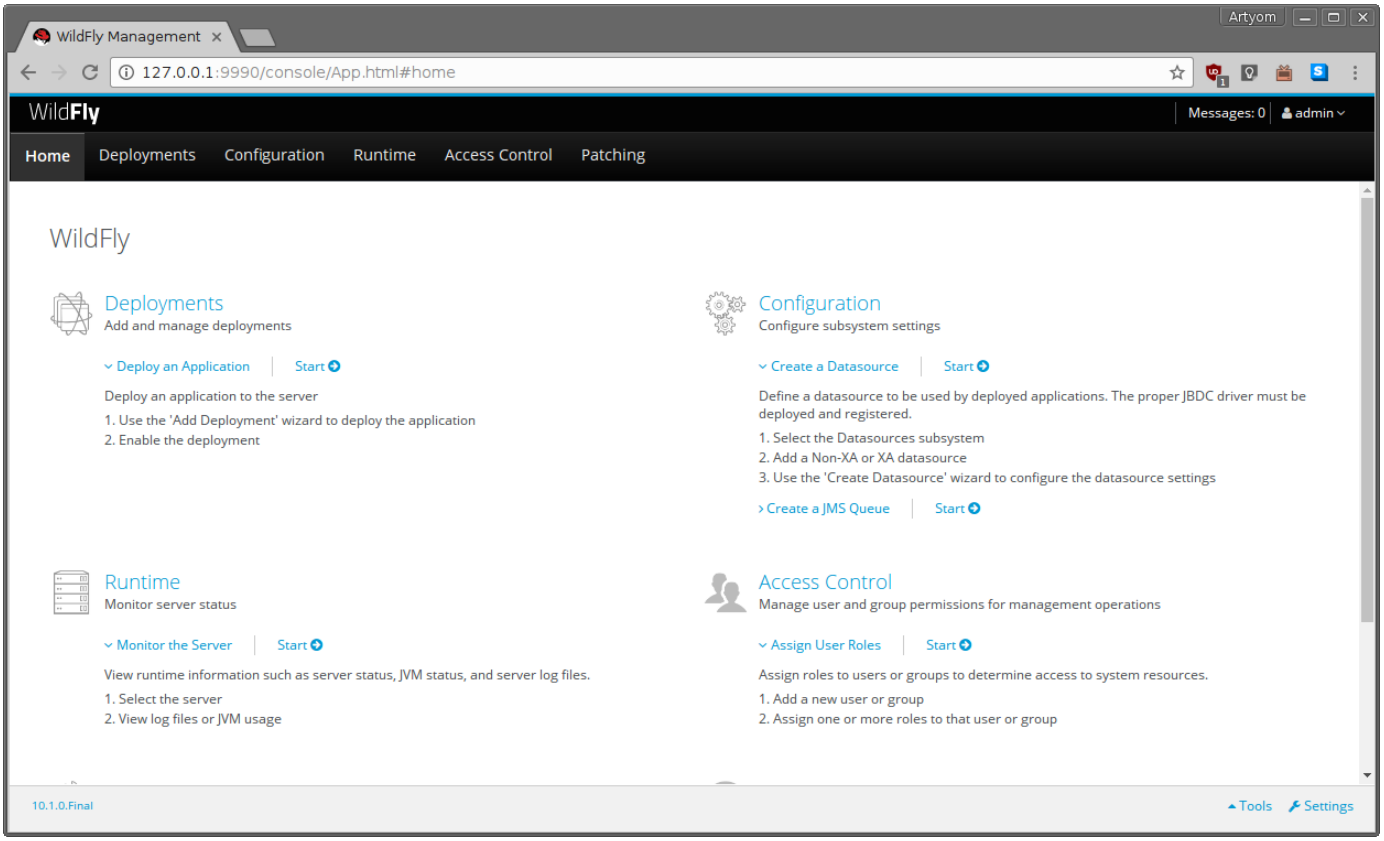

Рис. 13

В пункте Deployment выберем пункт Start:

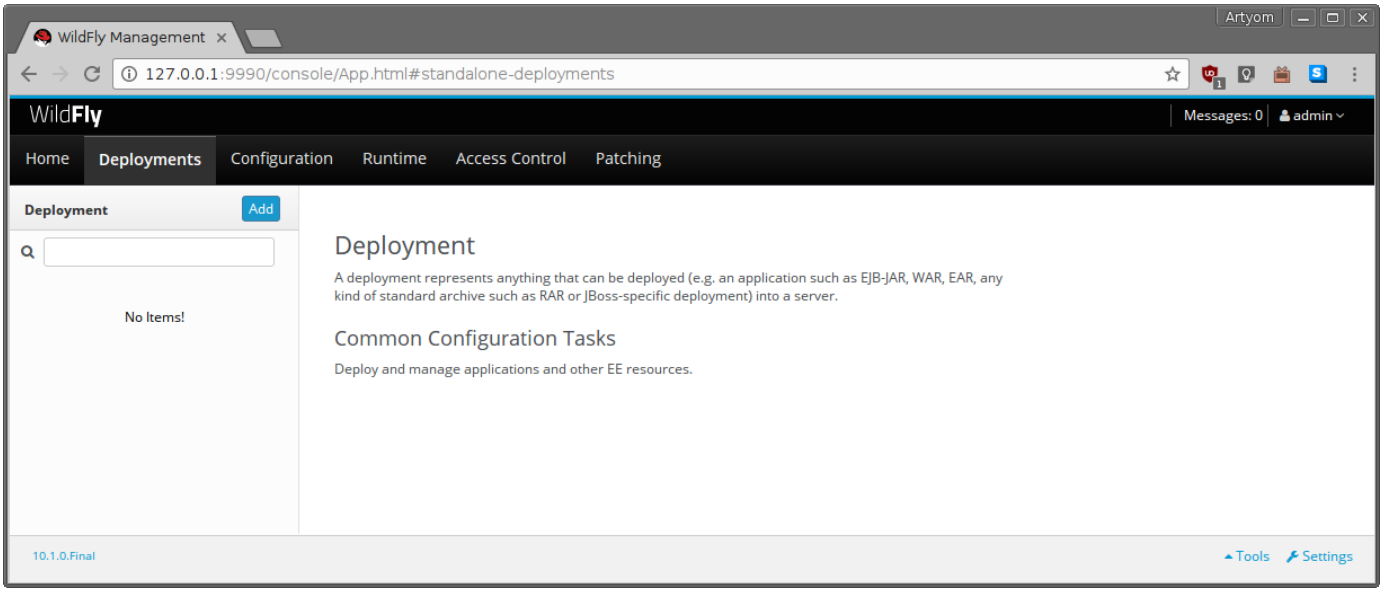

Рис. 14

Далее необходимо нажать на кнопку Add. Откроется окно, где нужно выбрать собранный war-файл:

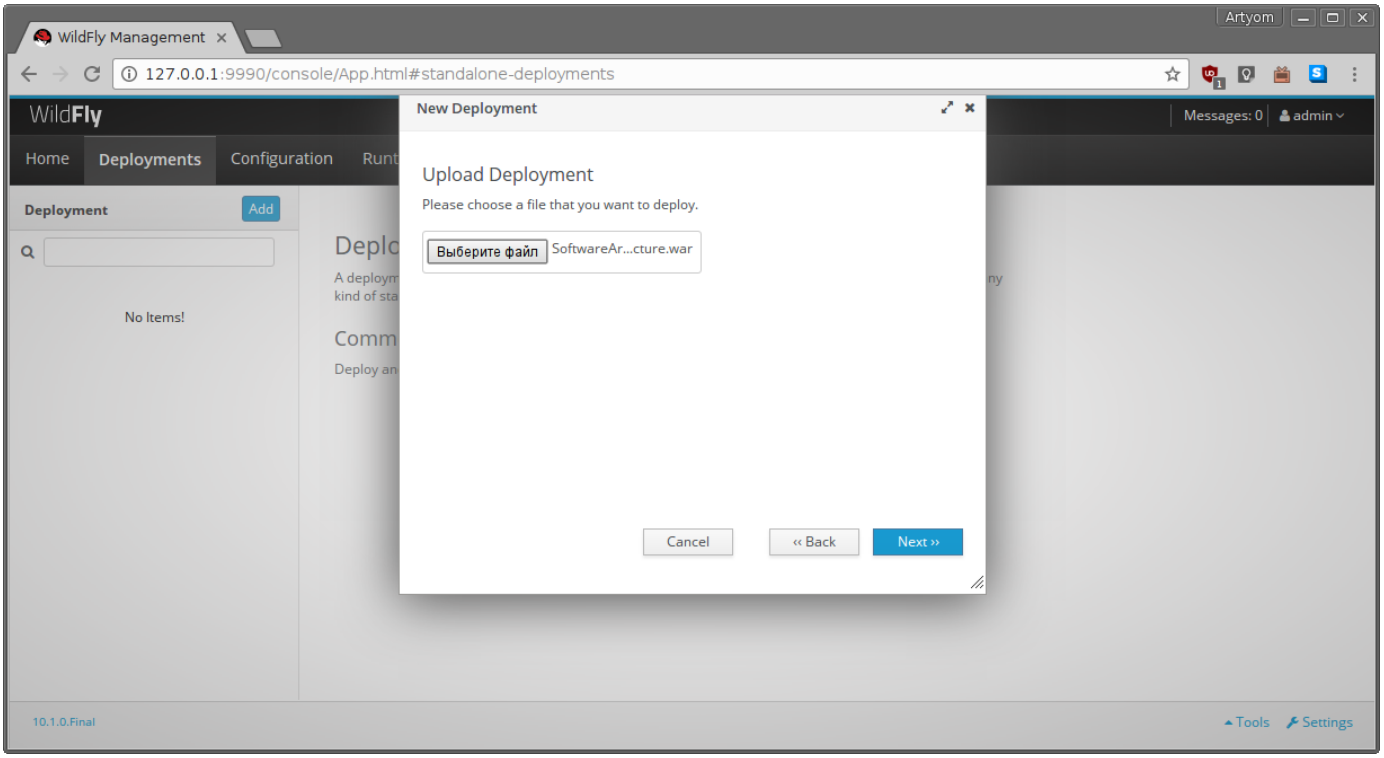

Рис. 15

Развернутое приложение доступно по следующему адресу: `http://127.0.0.1:8080/SoftwareArchitecture/`

5. Инструкция пользователя

По умолчанию веб-сайт доступен по адресу `http://127.0.0.1:8080/SoftwareArchitecture/`.

Ниже приведен снимок экрана с главной страницей веб-приложения.

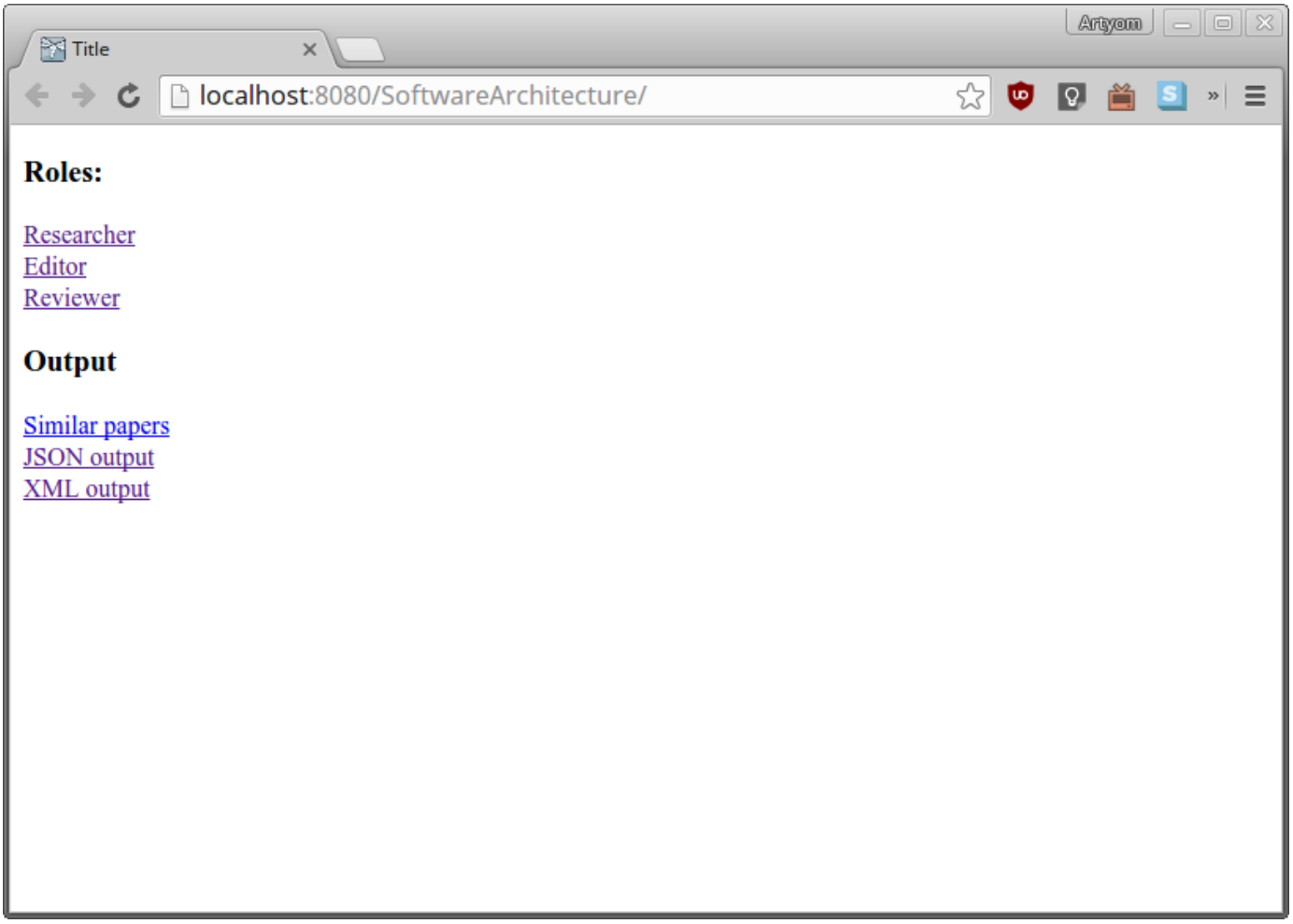

Рис. 16

5.1. Инструкция исследователя

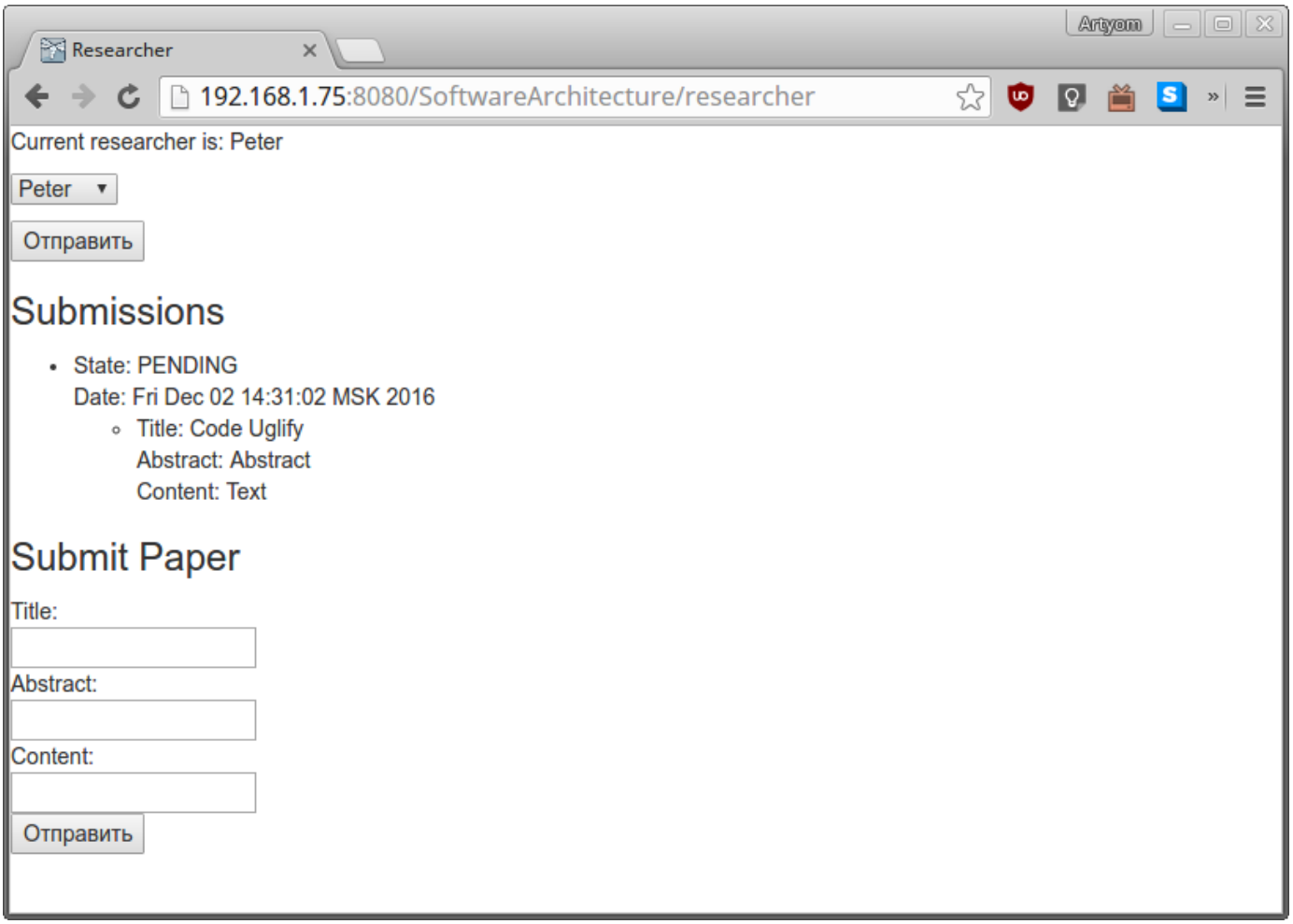

Рис. 17: Страница исследователя

Если пользователь - исследователь, ему необходимо пройти в раздел Researcher по соответствующей ссылке. Для дальнейшей работы пользователю необходимо аутентифицироваться. В верхней части страницы в выпадающем списке необходимо выбрать нужного пользователя и нажать кнопку «Отправить».

Следующий этап - заполнить на странице поля «Заголовок (Title)», «Реферат (Abstract)» и «Содержимое статьи (Content)». После заполнения всех полей следует нажать кнопку «Отправить».

После отправки статьи пользователь может увидеть её в списке в верхней части экрана.

5.2. Инструкция редактора

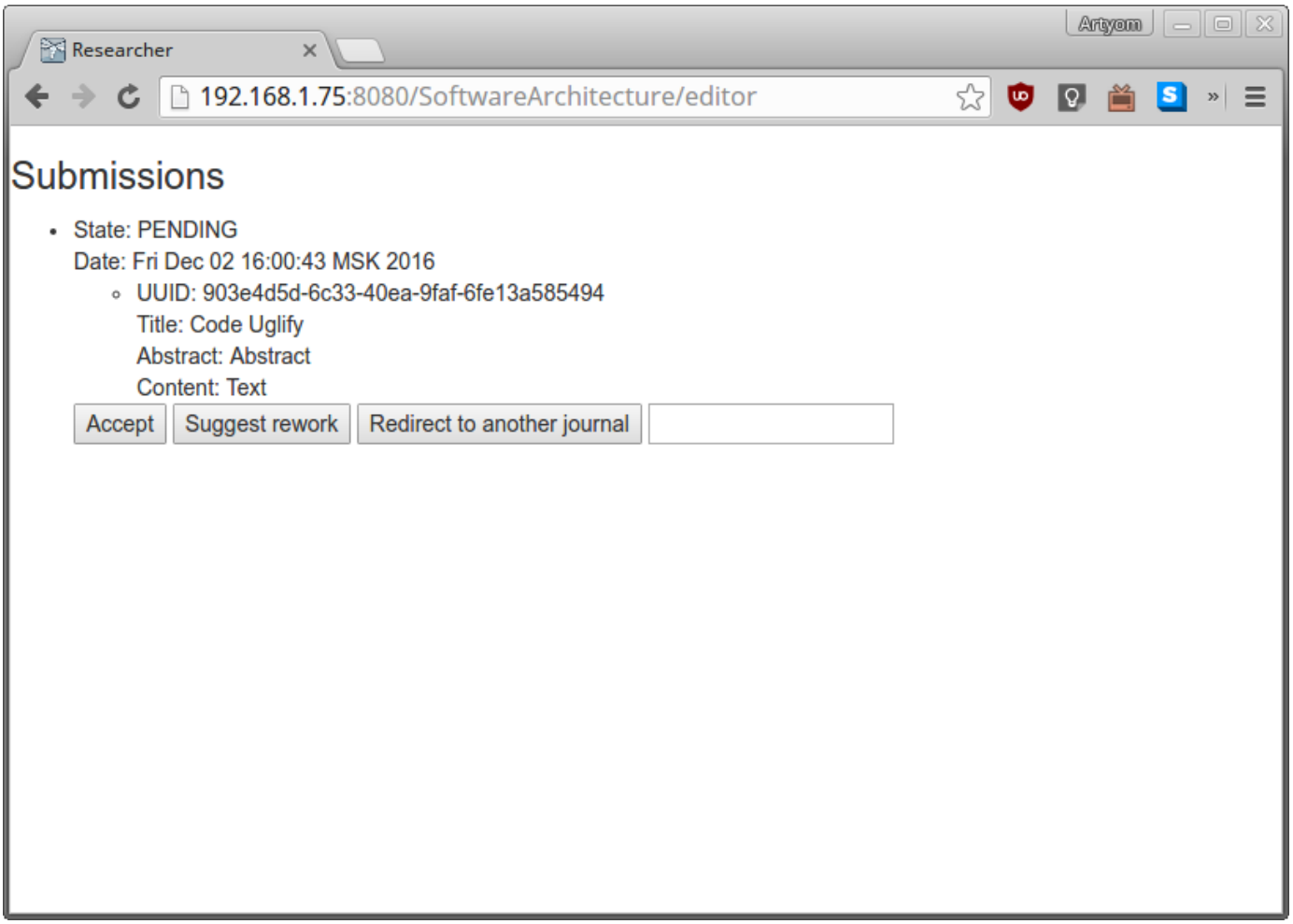

Рис. 18: Страница редактора

Если пользователь - редактор, ему необходимо пройти в раздел Editor по соответствующей ссылке. В этом разделе в верхней части страницы перечислены новые (ещё не просмотренные) статьи. В списке для каждой статьи указан её заголовок, реферат и непосредственно содержимое.

Если редактор решает, что статья соответствует тематике журнала и удовлетворяет правилам оформления, ему нужно нажать кнопку «Accept». Странице обновляется, и статья пропадает из списка не просмотренных статей. Если редактор отказывает в приеме статьи по причине недоработок в статье (например, проблемах с форматированием), ему необходимо указать список замечаний, который будет передан автору, и нажать кнопку «Suggest rework». Если редактор отказывает в приеме по причине несоответствия тематике журнала, он указывает список замечаний и нажимает кнопку «Redirect to another journal».

5.3. Инструкция рецензента

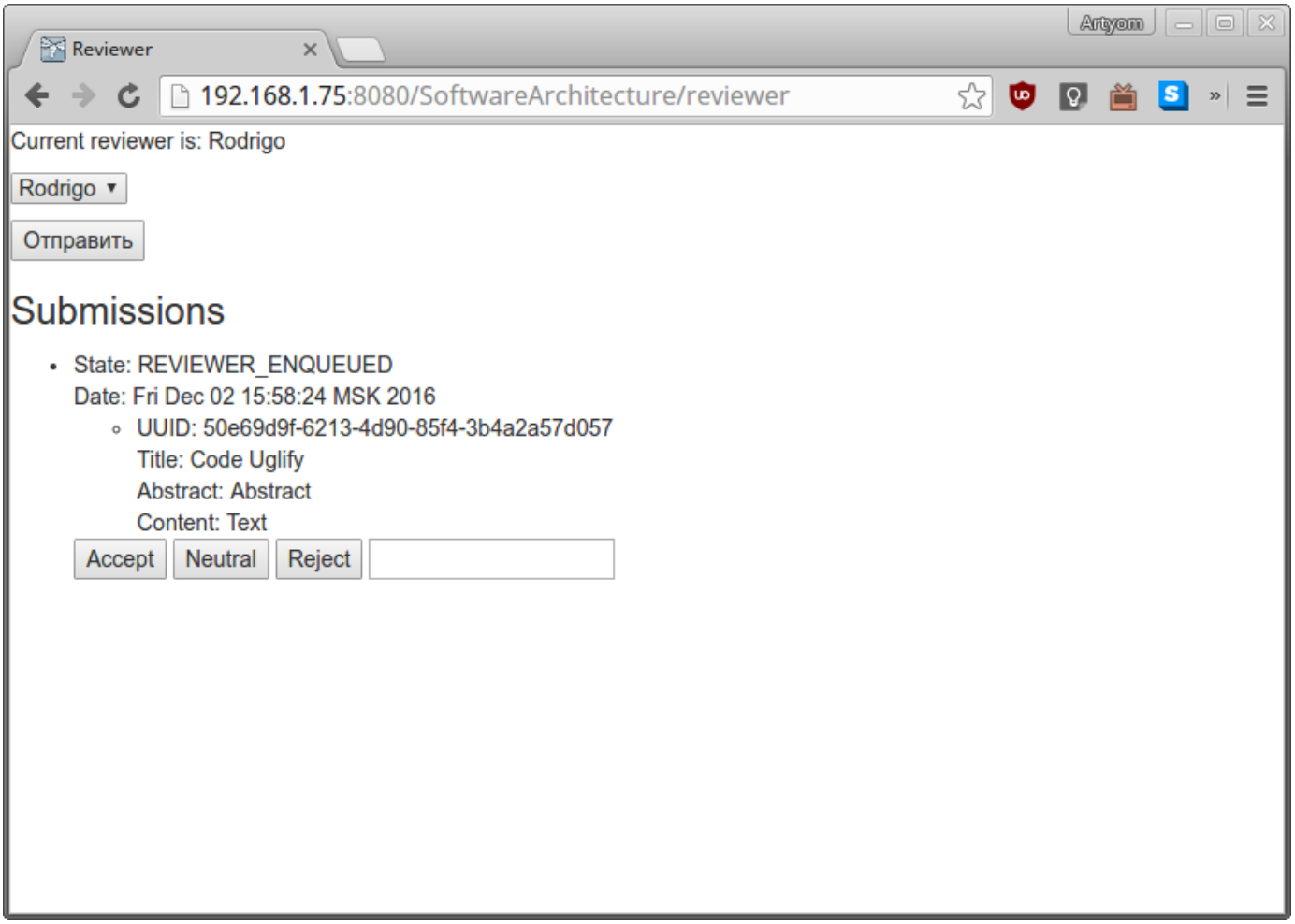

Рис. 19: Страница рецензента

Если пользователь - рецензент, ему необходимо пройти в раздел Reviewer по соответствующей ссылке. В верхней части страницы в выпадающем списке необходимо выбрать нужного пользователя и нажать кнопку «Отправить». Под блоком аутентификации на странице перечислены новые (ещё не просмотренные) статьи. В списке для каждой статьи указан её заголовок, реферат и непосредственно содержимое.

Если рецензент одобряет статью для публикации (Accept), то ему нужно нажать кнопку «Accept». Если рецензент имеет замечания к статье, но допускает её для публикации (Neutral), он должен нажать кнопку «Neutral». Если рецензент имеет замечания к статье и не допускает её для публикации (Reject), он должен нажать кнопку «Reject». Во всех случаях в текстовом поле около кнопок необходимо указать рецензию. После нажатия одной из кнопок статья отмечается в системе как принятая и исчезает из списка.

6. Выводы

В рамках данного курса были изучены принципы разработки программного обеспечения для распределенных вычислительных систем. Были изучены техно-

логии EJB (Enterprise Java Beans) и JPA (Java Persistence API). В соответствии с этими принципами и с использованием перечисленных технологий было разработано приложение - информационная система для научного журнала.

В качестве сервера приложений использовался WildFly 10, в качестве ORM - Hibernate 5. Для хранения данных использовалась СУБД PostgreSQL 9.5.

Возможные пути улучшения разработанного приложения:

- Добавление возможности составления и верстки журнала. При этом возможно появление ещё одной роли - верстальщика. Результат верстки - электронный документ в одном из распространенных форматов (например, PDF).
- Добавление интерфейса читателя

Приложение удовлетворяет требованиям прозрачности, масштабируемости и открытости, во многом благодаря использованию технологий HTML, EJB и JPA.