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, позволяющее продемонстрировать применение основных принципов проектирования программного обеспечения. В частности, в приложении необходимо было выделить следующие компоненты:

- Слой бизнес-логики
- Слой хранения данных
- Слой представления

Было решено разработать информационную систему для научного журнала.

1.1. 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 статьи

1) **Исследователь** выбирает **тему** и при поддержке научного руководителя проводит исследования

2) По результататам исследований автор при поддержке соавторов пишет **научную статью**

3) Автор выбирает **журнал** для публикации

4) Автор получает от редакторов журнала **правила оформления**

5) Автор редактирует статью в соответствии с правилами

    - Альтернатива: Редакция журнала сама готова привести **форматирование** к требуемому виду

2.2. Подача

1) Автор отправляет статью в **систему подачи статей**

2) **Редакция** просматривает статью

3) Статья передается на рецензирование

    - Альтернатива: Редакция отказывает в приеме статьи и отправляет исследователю **список замечаний**, чтобы он их исправил
    - Альтернатива: Редакция отказывает в приеме и рекомендует автору отправить статью в **другой журнал**

2.3. Рецензирование

1) Редакторы подбирают **рецензентов** и отправляют им статью

2) Рецензенты читают статью и составляют **отчет** по ней, содержащий **вопросы**, **замечания** и **общую оценку статьи**

3) Редакторы пересылают вопросы и замечания исследователю

4) В соответствии с оценкой редакция принимает **решение** о публикации

    - Альтернатива: Редакция отказывает в публикации, статья дорабатывается и передается либо на этап рецензирования, либо на этап подачи, либо в другой журнал

2.4. Подготовка и публикация

1) Редакция адаптирует статью под **макет журнала**

2) Редакция исправляет **ошибки правописания**

3) Статья помещается в **пул публикаций**, которые будут опубликованы в следующем **выпуске** журнала

   - Альтернатива: В дополнение к журналу, статья может быть **опубликована онлайн** индивидуально. Это происходит сразу же, без ожидания следующего выпуска журнала

4) Журнал из статей отправляется в **типографию**

   - Альтернатива: В дополнение к печатному варианту, журнал может быть опубликован в цифровом варианте

2.5. Диаграмма вариантов использования

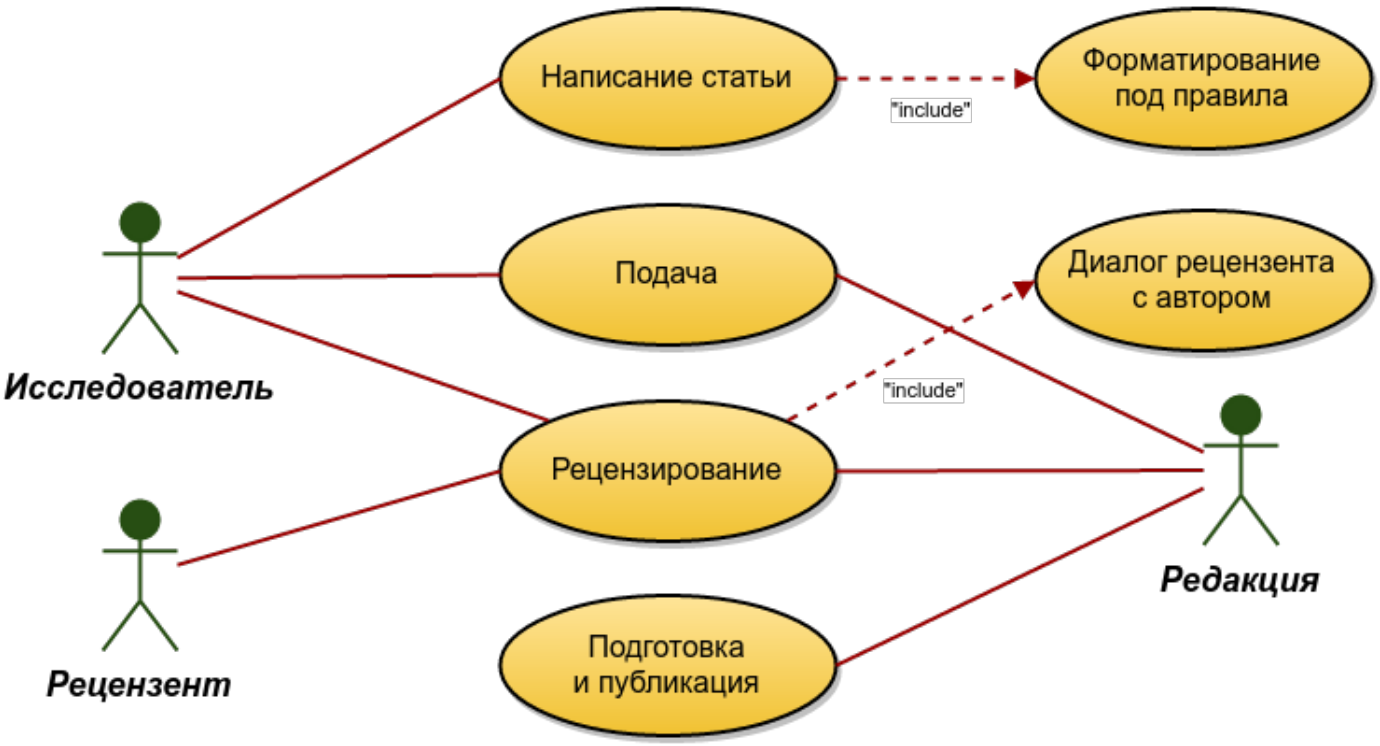

Рис. 1: Диаграмма вариантов использования

2.6. Модель предметной области

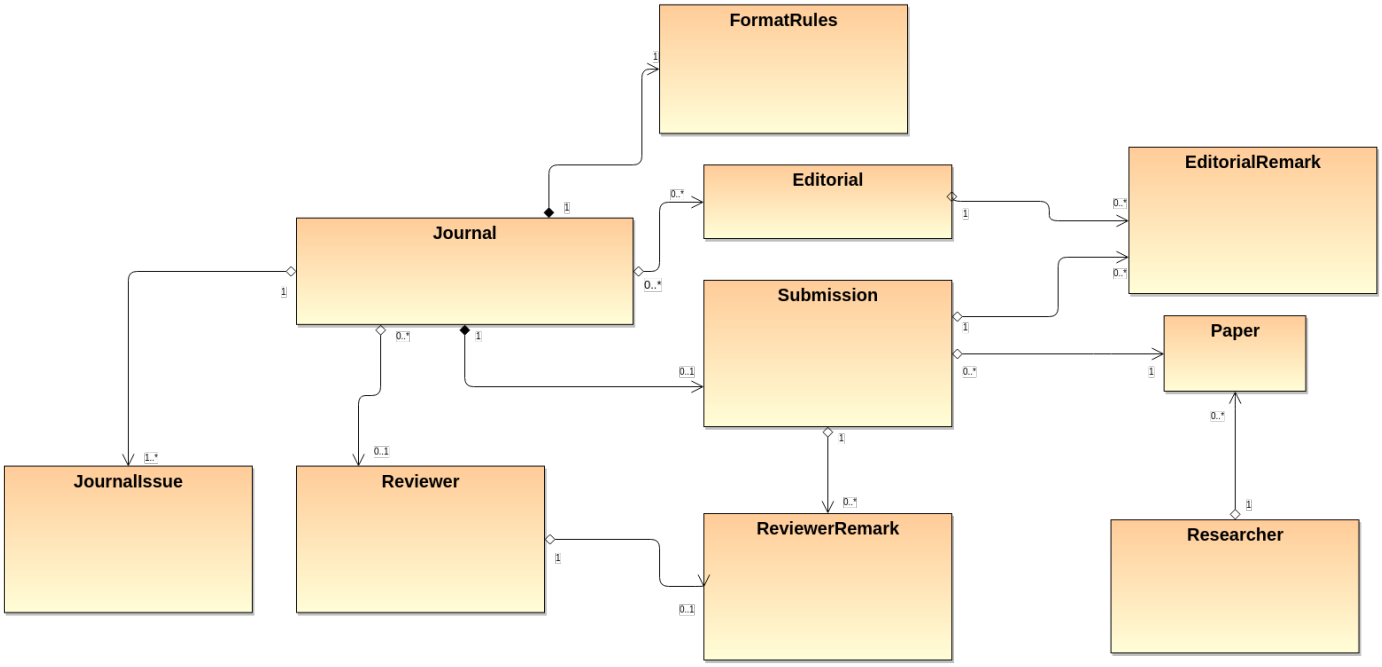

Рис. 2: Модель предметной области

2.7. Диаграмма последовательностей

2.7.1. Написание статьи и отправка в журнал

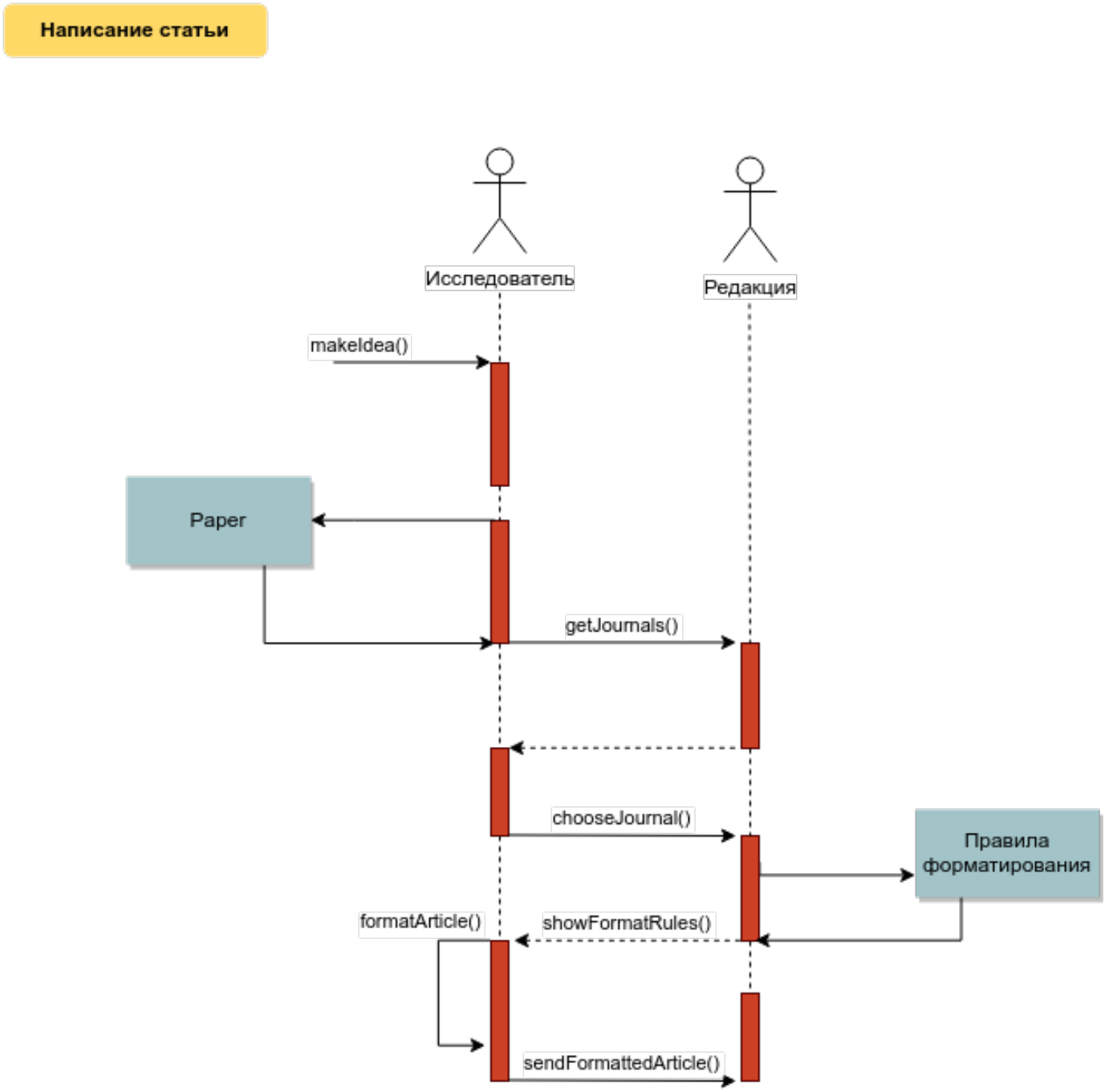

Рис. 3

2.7.2. Рецензирование

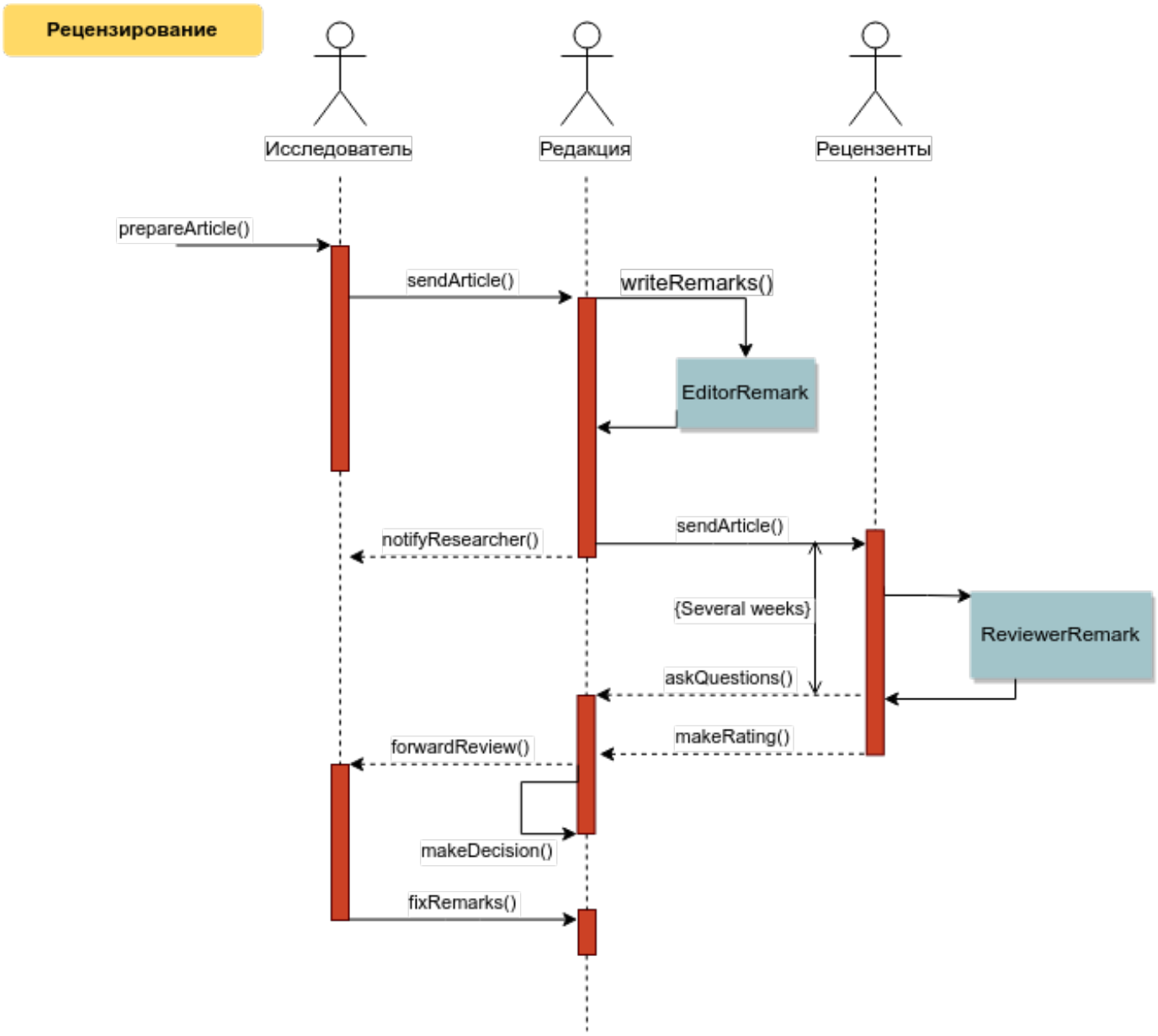

Рис. 4

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.

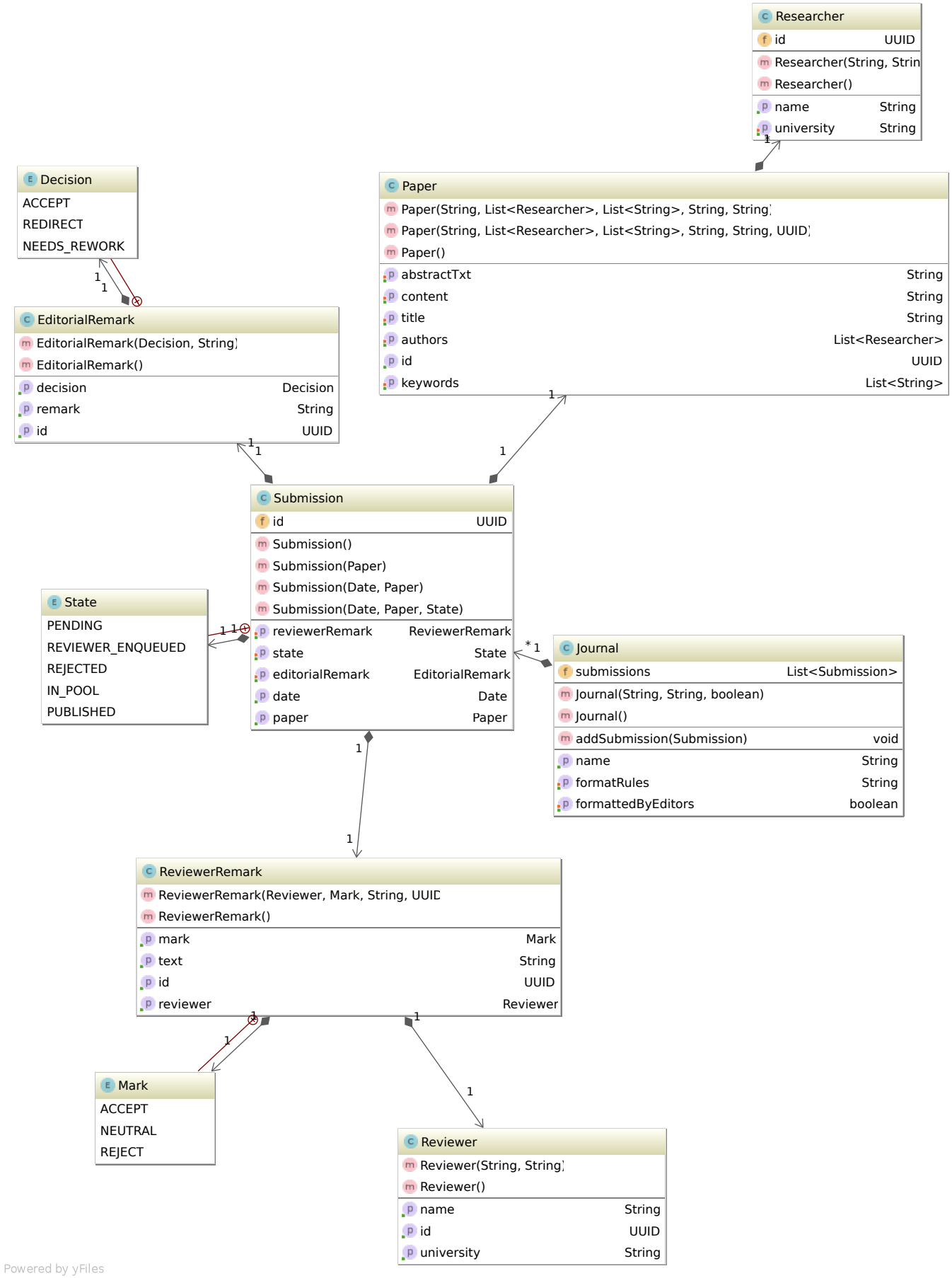

Рис. 5: Диаграмма класса для пакета objects

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

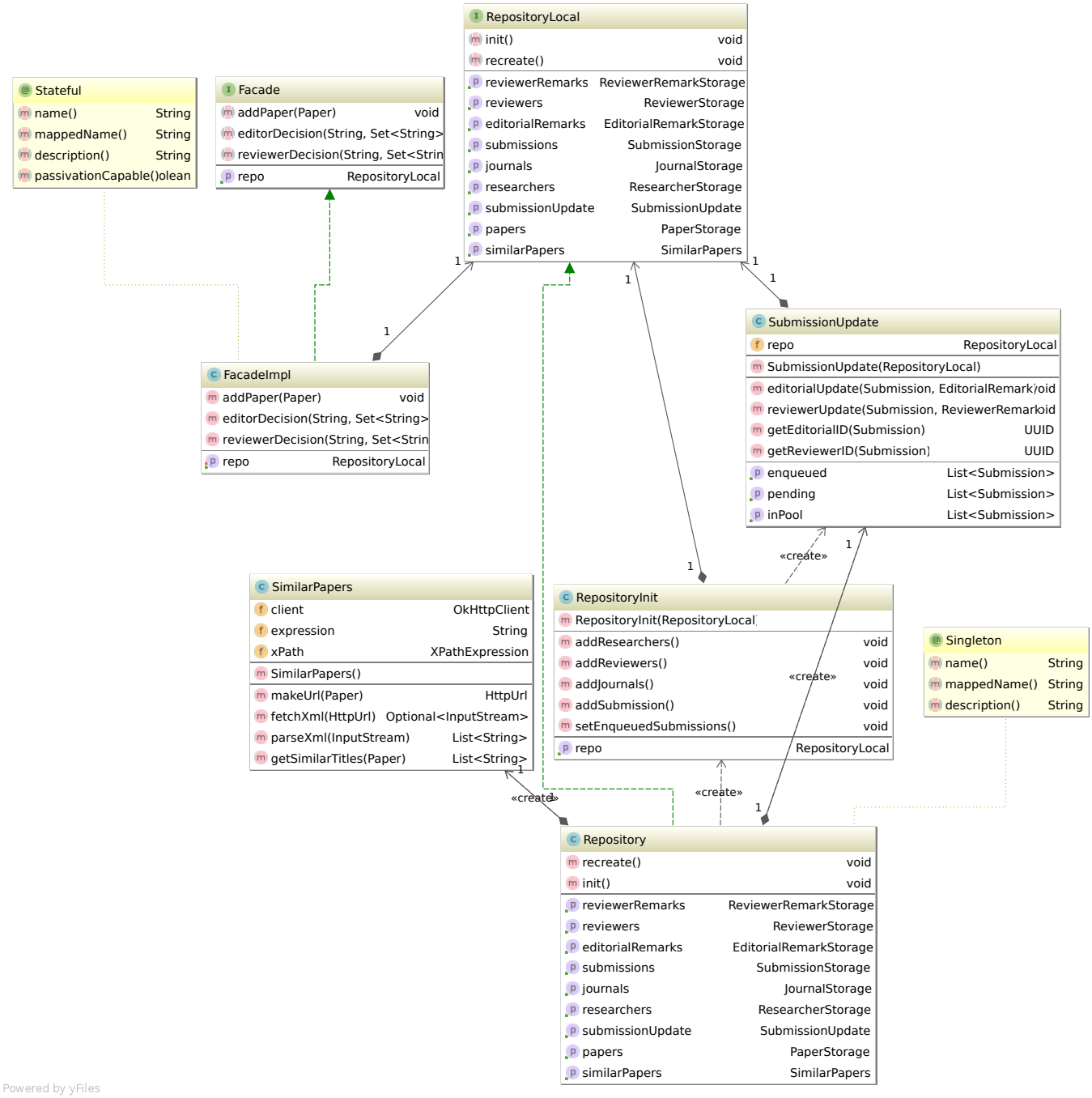

Рис. 6

На рисунке 7 показана диаграмма классов для слоя хранения.

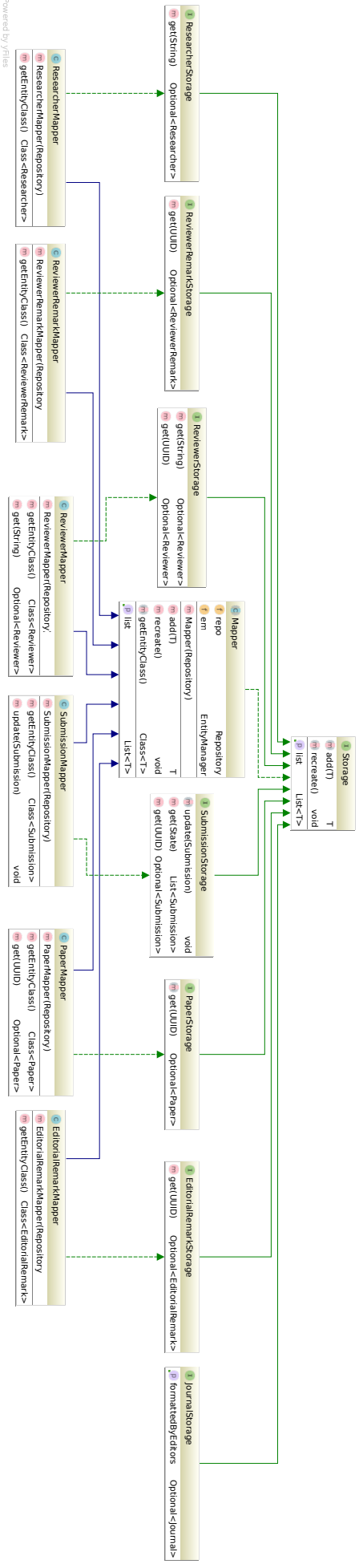
Powered by yFiles
Storage
add(T) T
recreate() void
list List<T>
ResearcherStorage
get(String) Optional<Researcher>
ReviewerRemarkStorage
get(UUID) Optional<ReviewerRemark>
ReviewerStorage
get(String) Optional<Reviewer>
get(UUID) Optional<Reviewer>
Mapper
repo Repository
em EntityManager
Mapper(Repository) 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>
updateSubmission() void
PaperMapper
PaperMapper(Repository)
getEntityClass() Class<Paper>
get(UUID) Optional<Paper>
EditorialRemarkMapper
EditorialRemarkMapper(Repository
getEntityClass() Class<EditorialRemark>

Рис. 7

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».
- 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

3.3. Инструкция системного администратора

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

BUILD SUCCESSFUL

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:

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

What type of user do you wish to add?
 a) Management User (mgmt-users.properties)
 b) Application User (application-users.properties)
(a):

Enter the details of the new user to add.
Using realm 'ManagementRealm' as discovered from the existing property files.
Username : user
Password recommendations are listed below. To modify these restrictions edit the add-user.properties configuration file.
 - The password should be different from the username
 - The password should not be one of the following restricted values {root, admin, administrator}
 - The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
WFLYDM0099: Password should have at least 8 characters!
Are you sure you want to use the password entered yes/no? yes
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[  ]:
About to add user 'user' for realm 'ManagementRealm'
Is this correct yes/no? yes
Added user 'user' to file '/home/artyom/Tools/wildfly-10.1.0.Final/standalone/configuration/mgmt-users.properties'
Added user 'user' to file '/home/artyom/Tools/wildfly-10.1.0.Final/domain/configuration/mgmt-users.properties'
Added user 'user' with groups  to file '/home/artyom/Tools/wildfly-10.1.0.Final/standalone/configuration/mgmt-groups.properties'
Added user 'user' with groups  to file '/home/artyom/Tools/wildfly-10.1.0.Final/domain/configuration/mgmt-groups.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
yes/no? no
```

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

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


  JBoss Bootstrap Environment

  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:

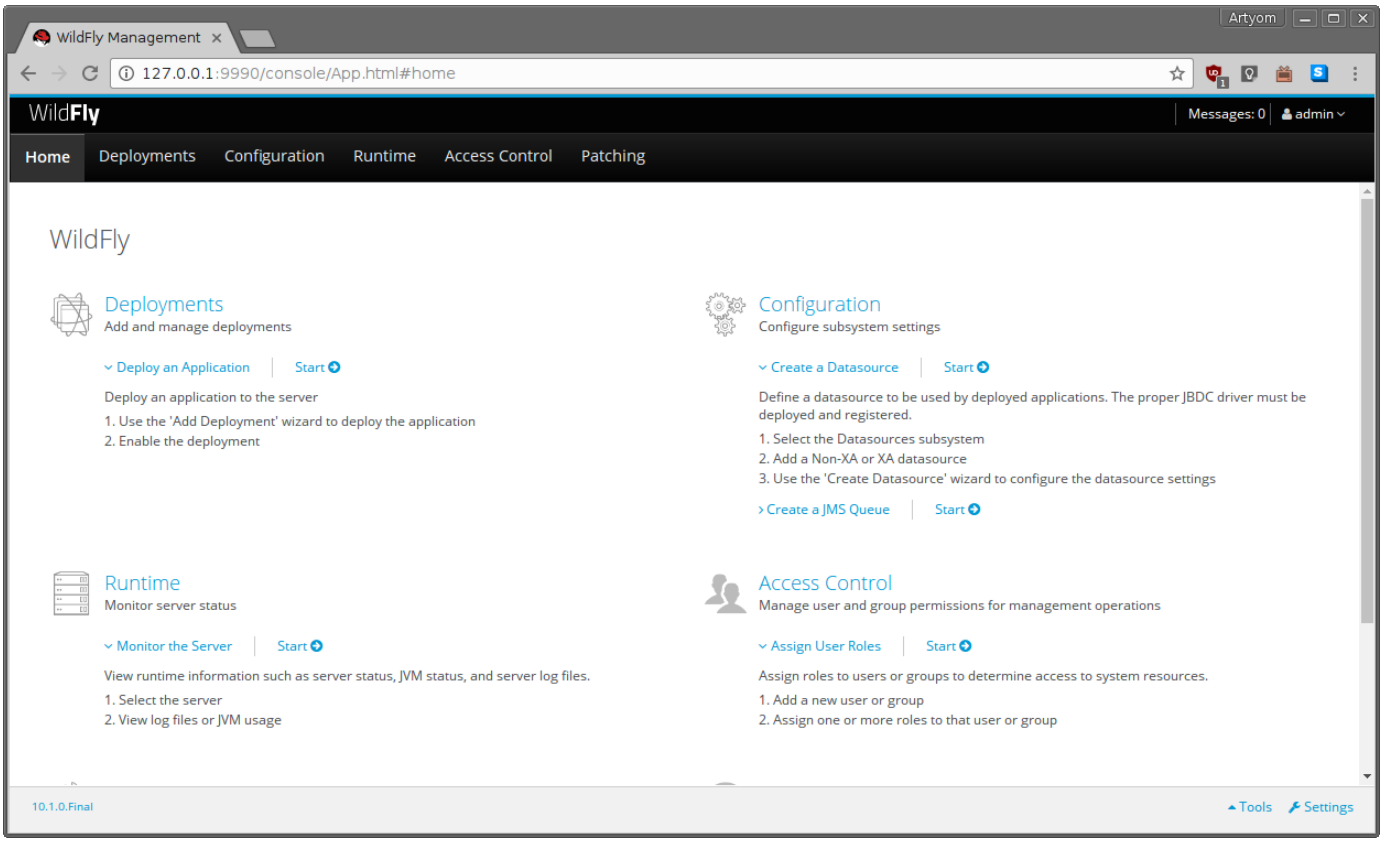

Рис. 8

В пункте Deployment выберем пункт Start:

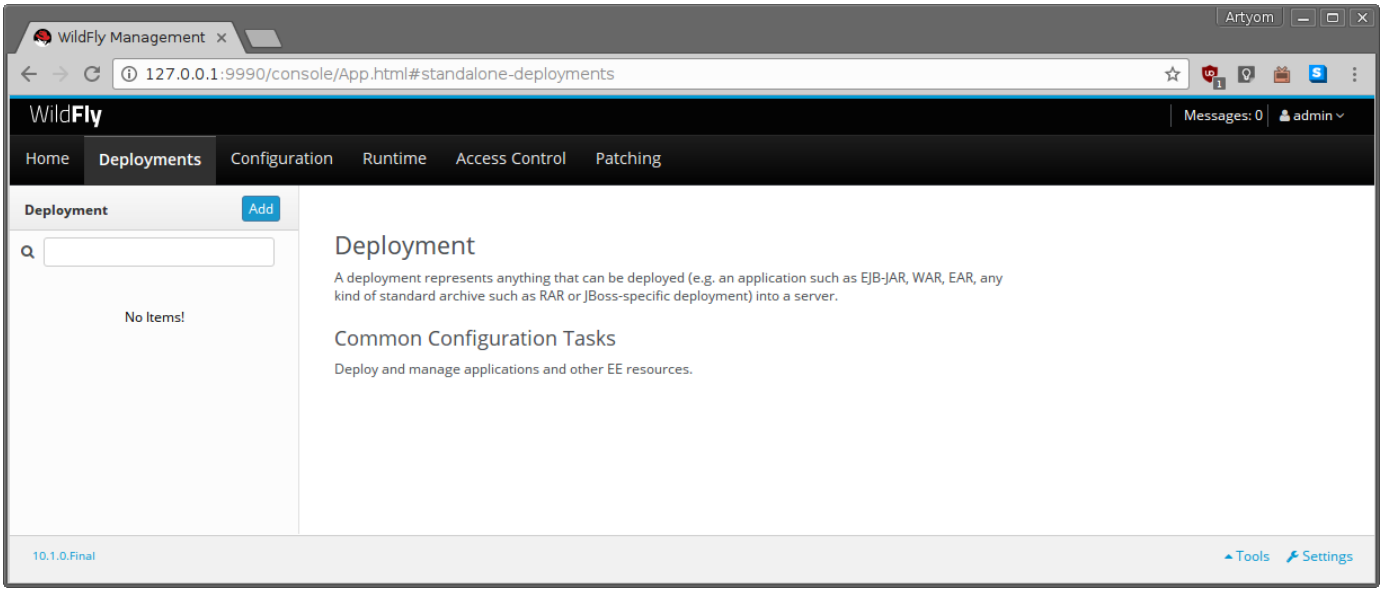

Рис. 9

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

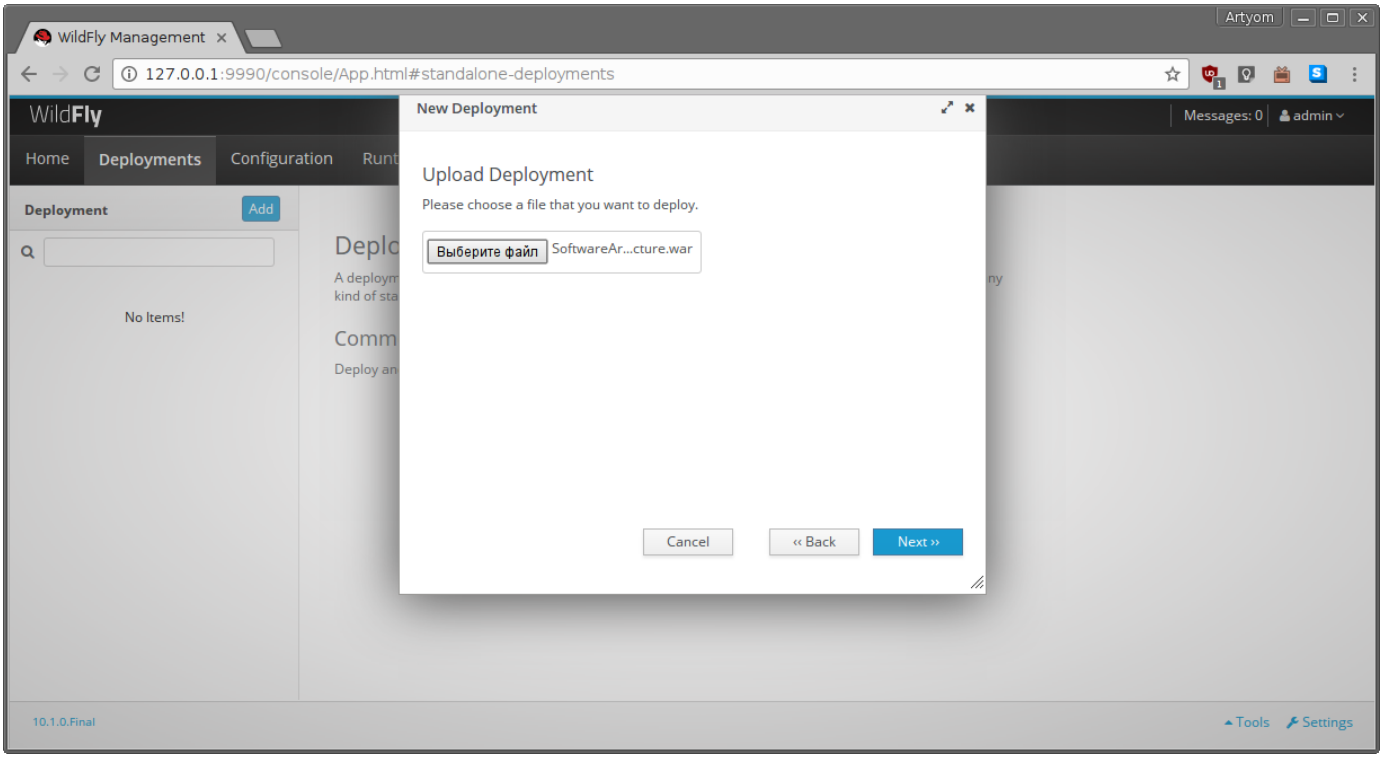

Рис. 10

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

4. Выводы

В рамках данного курса были изучены принципы разработки архитектуры программного обеспечения, а так же, следуя этим принципам, было разработано при-

ложение. в приложении было создано три слоя:

- Слой бизнес-логики
- Слой хранения данных
- Слой представления

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