Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчет по курсовой работе

Дисциплина «Программное обеспечение распределенных вычислительных систем»

Работу выполнил студент группы 63501/3
_____ Хандыго Е.Д.

Работу приняла
_____ Стручков И.В.

Санкт-Петербург
2016 г.

Содержание

1 Анализ задания

В рамках данного курса необходимо разработать простое приложение, позволяющее продемонстрировать основные архитектурные принципы проектирования программного обеспечения с использованием технологии Enterprise Java Beans (EJB). В качестве системы рассмотрим систему управлению персоналом, позволяющую автоматизировать некоторые аспекты повседневной деятельности работников компании. На этапе проектирования такой системы были выделены следующие роли и их интересы:

- Работник
  - Сообщать время своего отсутствия в офисе.
  - Резервировать время отпуска.
- Менеджер
  - Имеет те же интересы, что и работник.
  - Одобрять/отклонять запросы закрепленных за ним работников.
  - Выписывать премию любому закрепленному за ним работнику.
- Бухгалтер
  - Имеет те же интересы, что и работник.
  - Одобрять/отклонять премию, выписанную работнику, закрепленному за ним.

Используя эти данные, построим модель вариантов использования системы в виде UML диаграмы, как показано на рисунке 1. При этом в системе должны функционировать следующие бизнес процессы:

- Запрос на предоставление отпуска:
  1. Пользователь системы выбирает желаемый период отпуска.
  2. Если менеджер не имеет вышестоящего начальника, то запрос автоматически подтверждается системой. В противном случае см. пункт 3.
  3. Запрос поступает менеджеру, за которым закреплен работник.
  4. Менеджер принимает или отклоняет запрос пользователя на предоставление отпуска в указанный период.
  5. В случае утверждения отпуска, соответствующая запись появляется в рабочем плане работника.
- Премирование работника:
  1. Менеджер формирует запрос на выплату премии работнику, который закреплен за ним.
  2. Запрос поступает бухгалтеру, за которым закреплен данный работник.
  3. Бухгалтер утверждает или отклоняет запрос на выплату премии.

4. В случае утверждения премии, соответствующая запись появляется в списке премиальных выплат работника.

- Указание времени отсутствия:

1. Работник может указать период собственного отсутствия по какой-либо причине.
2. Данное действие не требует подтверждения со стороны — соответствующая запись сразу же размещается в рабочем плане работника.
3. Работник может удалить любую такую запись из своего собственного рабочего плана.

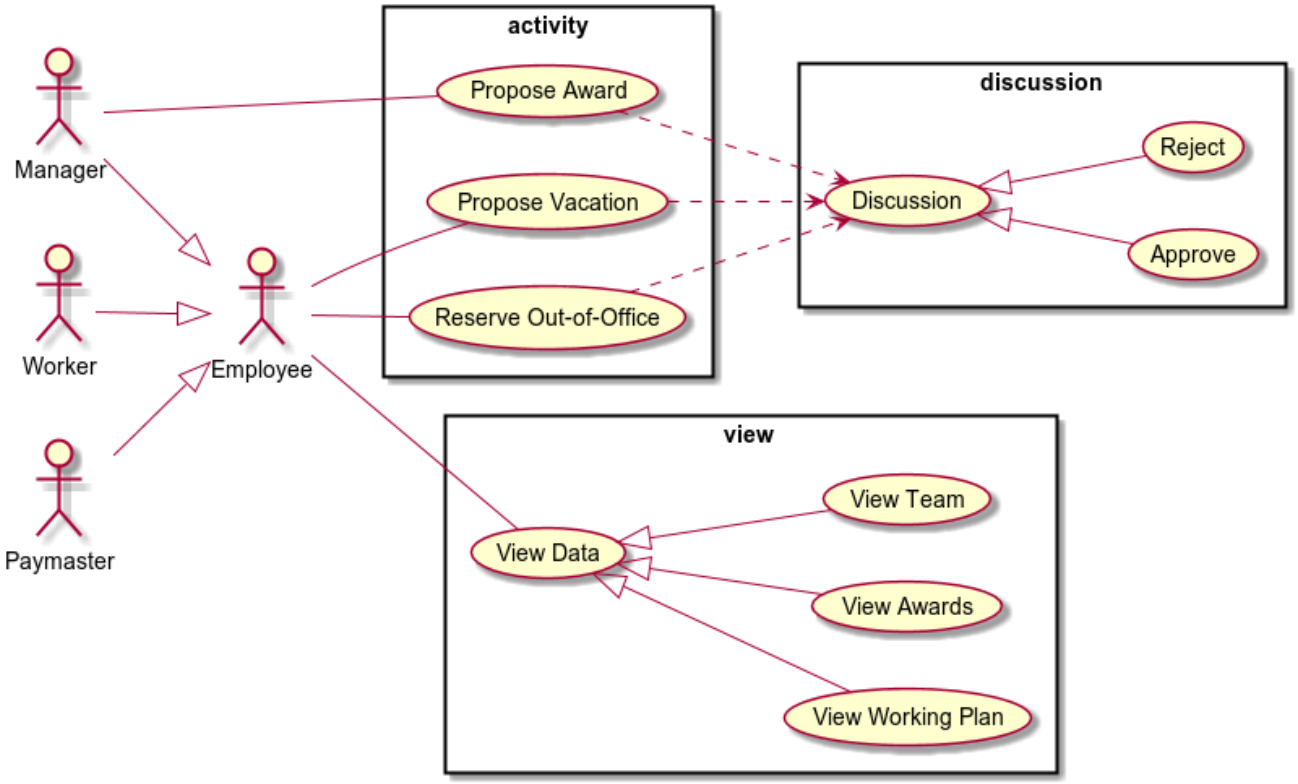

Рис. 1: Диаграма вариантов использования системы

Приведем здесь подробный разбор вариантов использования системы:

- Заявка на выделение премии:

1. **Пользователь**, выбирает сотрудника, которому необходимо выписать премию.
2. **Система** открывает форму премирования.
3. **Пользователь** заполняет данную форму, указывая **когда** и в каком **размере** должна быть выплачена премия.
4. **Система** проверяет, что пользователь, отправивший заявку, действительно является менеджером по отношению к премируемому сотруднику.
5. В случае успеха **система** заносит запись о созданной заявке в базу данных.

6. В случае ошибки на любом этапе **система** выводит сообщение об ошибке.

- Заявка на предоставление отпуска:

1. **Пользователь** нажимает кнопку.
2. **Система** открывает форму заявки на предоставление отпуска.
3. **Пользователь** заполняет форму.
4. **Система** проверяет, что пользователь, отправивший заявку, соответствует работнику, на имя которого запрашивается отпуск.
5. В случае успеха **система** заносит запись о созданной заявке в базу данных.
6. В случае ошибки на любом этапе **система** выводит сообщение об ошибке.

- Резервирование времени вне офиса:

1. **Пользователь** нажимает кнопку.
2. **Система** открывает форму заявки на резервацию времени вне офиса.
3. **Пользователь** заполняет форму.
4. **Система** проверяет, что пользователь, отправивший заявку, соответствует работнику, на имя которого отправляется заявка.
5. В случае успеха **система** заносит запись о созданной заявке в базу данных.
6. В случае ошибки на любом этапе **система** выводит сообщение об ошибке.

- Подтверждение заявки:

1. **Пользователь** нажимает кнопку.
2. **Система** проверяет, что роль пользователя соответствует выполняемой операции:
   - Если подтверждается заявка на премию, то пользователь должен иметь роль «бухгалтер».
   - Если подтверждается заявка на предоставление отпуска, то пользователь должен иметь роль «менеджер».
3. В случае успеха **система** обновляет запись о соответствующей заявке в базу данных.
4. В случае ошибки на любом этапе **система** выводит сообщение об ошибке.

- Отклонение заявки:

1. **Пользователь** нажимает кнопку.
2. **Система** проверяет, что роль пользователя соответствует выполняемой операции:
   - Если отклоняется заявка на премию, то пользователь должен иметь роль «бухгалтер».
   - Если отклоняется заявка на предоставление отпуска, то пользователь должен иметь роль «менеджер».

3. В случае успеха **система** обновляет запись о соответствующей заявке в базу данных.
4. В случае ошибки на любом этапе **система** выводит сообщение об ошибке.

Доменная модель системы в нашем случае описывается сущностями, представляющими пользователей системы и сущности, с которыми они взаимодействуют. Таким образом, модель предметной области в нотации UML будет выглядеть как показано на рисунке 2.

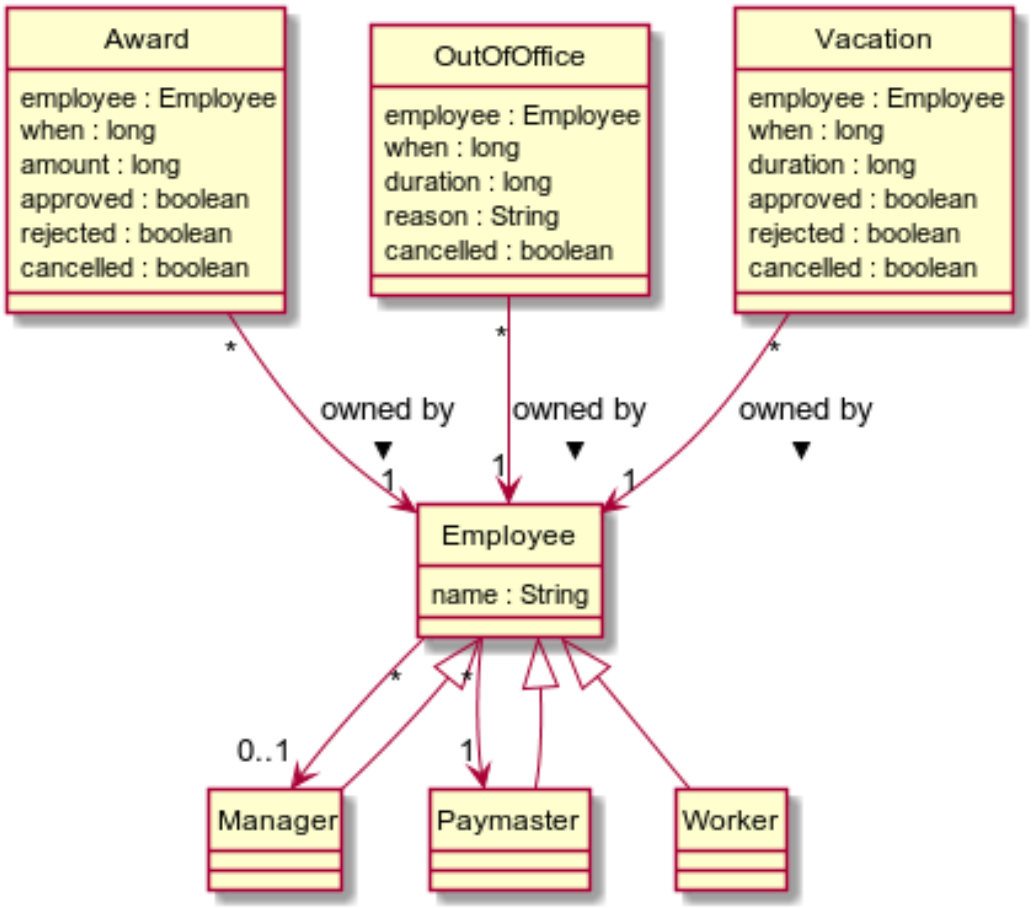

Рис. 2: Модель предметной области системы

2 Реализация

2.1 Объектно-ориентированное проектирование

Перед началом реализации необходимо провести объектно-ориентированное проектирование системы. Данный этап подразумевает представление бизнес логики в виде программных функций и разнесение этих функций по классам. Таким образом в программе формируется слой бизнес-логики. В данной случае этот слой было решено реализовать в виде набора сервисов. Наполнение классов осуществлялось так, чтобы соблюдались основные принципы проектирования, такие, например, как принцип единственной ответственности (single responsibility principle). Результаты проектирования представлены ниже, на рисунке 3 в виде UML диаграммы.

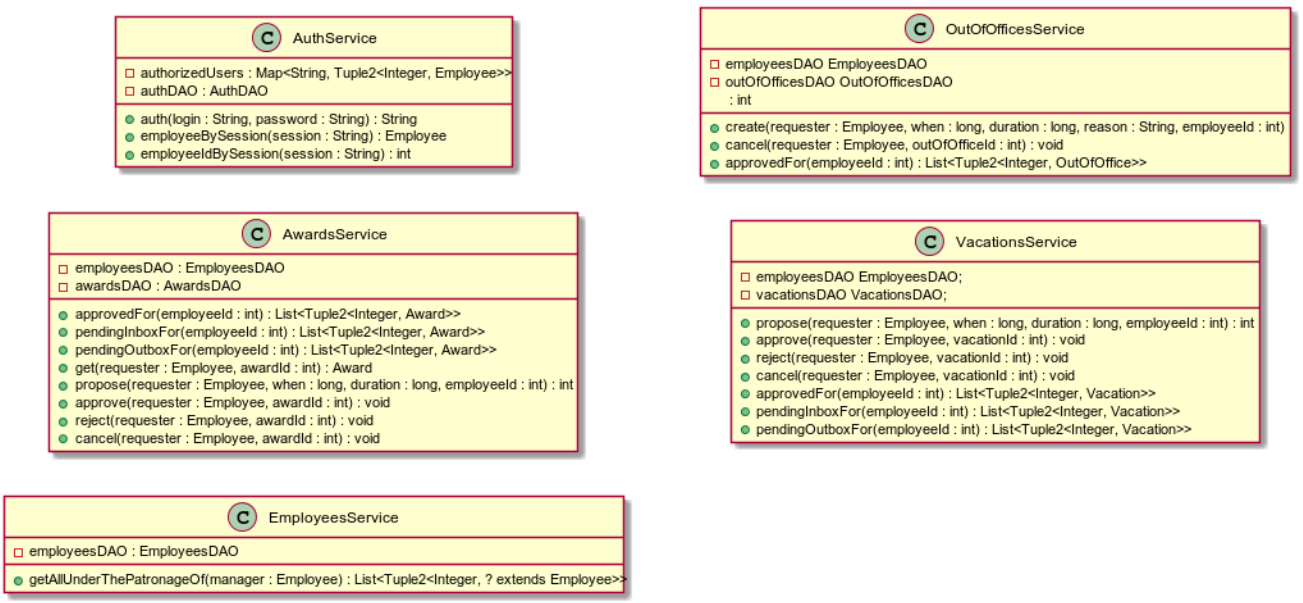

Рис. 3: Модель предметной области системы

Как видно, в результате проектирования было выделено 5 сервисов. Каждый из классов параметризуется только лишь объектами доступа к данным (DAO — data access object) соответствующего типа. Приведем краткий разбор данных классов:

- `AuthService` отвечает за авторизацию и аутентификацию пользователей, а также за выделение им специальных сессионных ключей.
  - Процедура `auth` отвечает за выделение и сохранение сессионного ключа пользователю с заданными логином и паролем.
  - Процедура `employeeBySession` отвечает за выдачу сущности пользователя по сессионному ключу.
  - Процедура `employeeIdBySession` отвечает за выдачу идентификационного номера сущности пользователя по сессионному ключу.
- `AwardsService` предназначен для работы с премиями.

  - Процедура `approvedFor` предназначена для выдачи премий заданного пользователя, которые уже были подтверждены бухгалтером.
  - Процедура `pendingInboxFor` предназначена для выдачи запросов на выплату премий, направленных заданному пользователю с ролью «бухгалтер».
  - Процедура `pendingOutboxFor` предназначена для для выдачи запросов на выплату премий, отправленных заданным пользователем с ролью «менеджер».
  - Процедура `get` выдает сущность-премию по ее идентификационному номеру. При этом осуществляется проверка прав доступа пользователя `requester` на доступ к заданной сущности.
  - Процедура `propose` предназначена для создания запроса на выдачу премии заданному пользователю в заданный срок и с заданным размером. При этом осуществляется проверка прав доступа пользователя `requester` на создание такого запроса с указанным получателем.
  - Процедура `approve` предназначена для подтверждения заданного запроса на выдачу премии. При этом осуществляется проверка прав доступа пользователя `requester` на выполнение данного действия.
  - Процедура `reject` предназначена для отклонения заданного запроса на выдачу премии. При этом осуществляется проверка прав доступа пользователя `requester` на выполнение данного действия.
  - Процедура `cancel` отменяет запрос на выдачу премий, который еще не был ни подтвержден, ни отклонен. При этом осуществляется проверка прав доступа пользователя `requester` на выполнение данного действия.

- `OutOfOfficeService` предназначен для работы с отсутствиями на рабочем месте.
  - Процедура `create` предназначена для резервирования времени отсутствия в офисе в заданный период и с указанной причиной. При этом осуществляется проверка прав доступа пользователя `requester` на создание такого запроса с указанным получателем.
  - Процедура `cancel` предназначена для отмены резервации времени нахождения вне офиса. При этом осуществляется проверка прав доступа пользователя `requester` на выполнение данного действия.
  - Процедура `approvedFor` предназначена для извлечения всех резерваций времени на отсутствие в офисе для указанного пользователя.

- `VocationService` предназначен для работы с отпусками. Здесь мы не будем приводить более подробный разбор методов данного класса, поскольку все они аналогичны уже рассмотренным методам из класса `AwardsService`.

- EmployeeService отвечает за работу с пользователями.
  - Процедура `getAllUnderThePatronageOf` отвечает за выдачу пользователей, которые подотчетны указанному пользователю.

Также в рамках этапе объектно-ориентированного проектирования был создан набор диаграмм последовательностей в нотации UML (sequence duagram). Ниже, на рисунках 4 – 6 представлено несколько таких диаграмм для операций создания, подтверждения и отклонения заявки на выдачу премии соответственно.

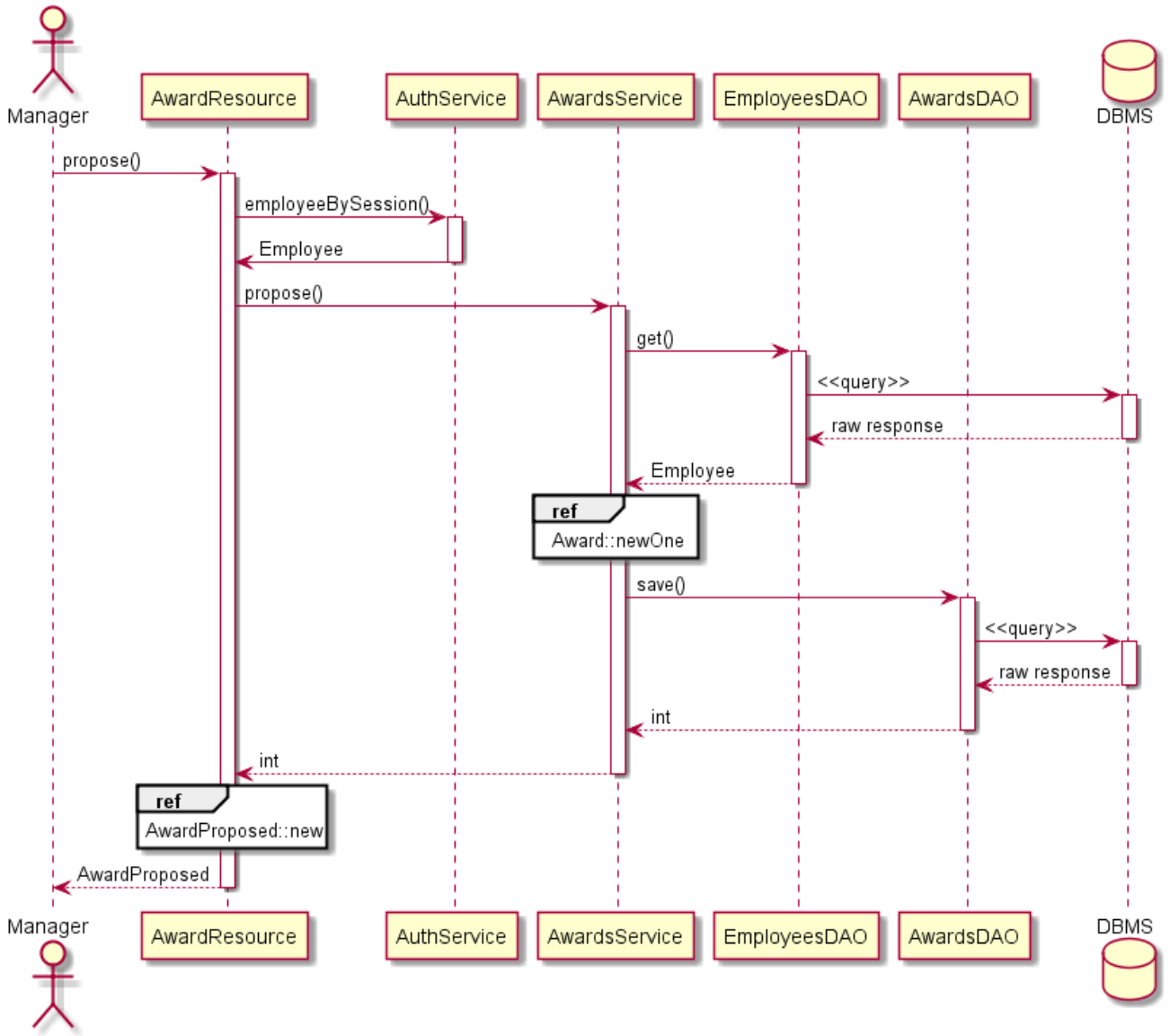

Рис. 4: Процесс создания заявки на премирование

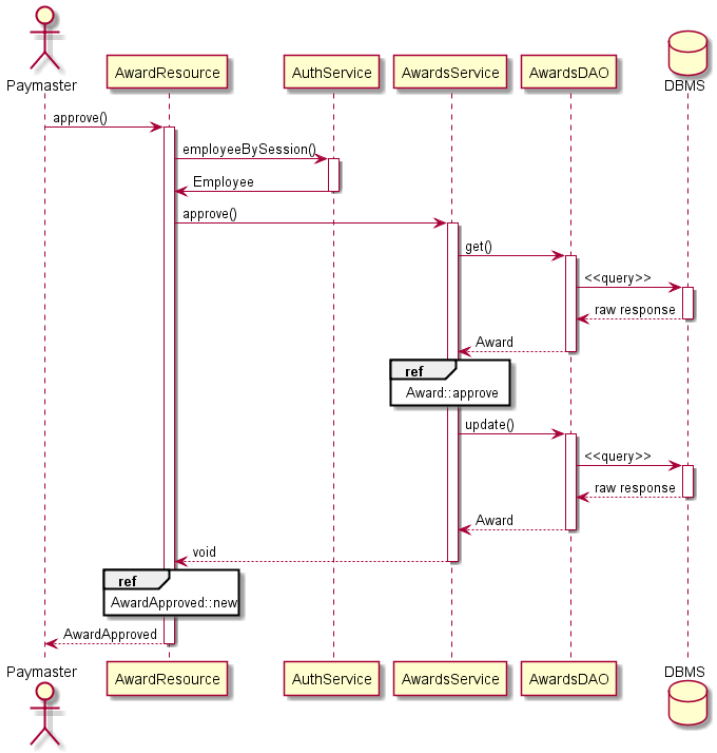

Рис. 5: Процесс подтверждения заявки на премирование

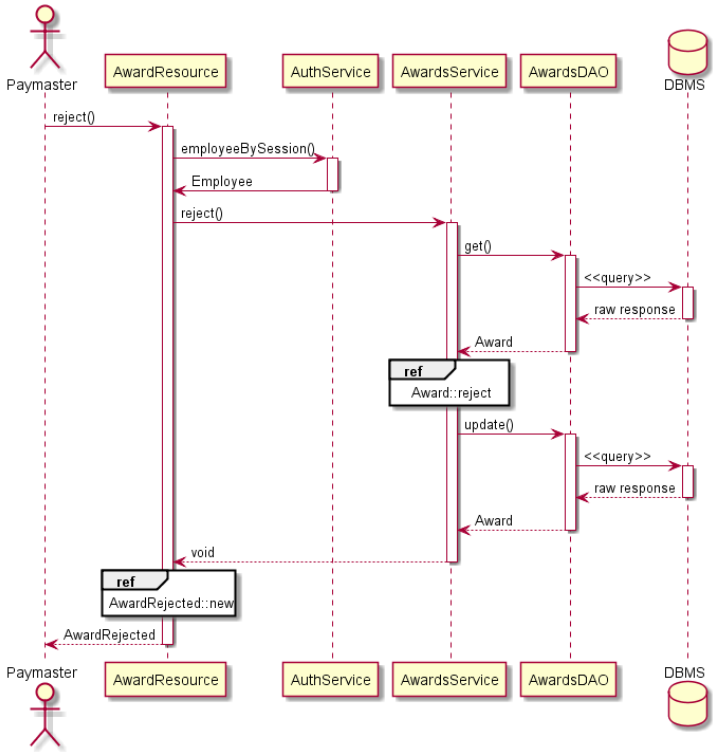

Рис. 6: Процесс отклонения заявки на премирование

2.2 Детали реализации

Разработанное приложение необходимо реализовать в виде сетевого (web based) приложения. По этой причине, помимо уровня бизнес логики был также выделен уровень представления приложения, отвечающий за принятие запросов от клиента и отправку ему ответов. Методы, реализуемые на данном уровне, работают почти одинаково по следующей схеме:

- Аутентификация пользователя путем обращения к экземпляру класса `AuthService`.
- Преобразование аргументов сетевого запроса к виду, удобному для уровня бизнес логики.
- Преобразованный запрос направляется на уровень бизнес логики, в соответствующий сервис.
- Результат, полученный от сервиса преобразуется к виду, удобному для ответа клиенту.

Такая логика работы позволяет обеспечить независимость уровня бизнес логики от уровня представлений и, как следствие, большую гибкость. Другими словами, сервисные классы не имеют понятия о том, что приложение является сетевым, что позволяет переиспользовать данный код при разработке, например, обычного (desktop) приложения. В то же время, любые изменения на уровне представления не затрагивают сервисный уровень и, следовательно, не могут его «сломать». Классы на уровне представления реализованы с учетом методики REST. Данный предоставляются в формате json. Преобразование объектов в формат json осуществляется при помощи библиотеки `jackson`.

Из рисунка 3 видно, что на уровне бизнес логики используются специальный объекты доступа к данным. Данные объекты также располагаются на отдельном уровне приложения — уровне доступа к данным. Принцип работы методов, расположенных на данном уровне, совпадает с логикой работы методов на уровне представления с той лишь только разницей, что здесь опускается этап аутентификации пользователей. Таким образом, обеспечивается такая же гибкость взаимоотношения между уровнями бизнес логики и доступа к данным, как и в случае уровней представления данных и бизнес логики. Для реализации данного уровня использовалась библиотека `ebean-orm`, осуществляющая преобразование сущностей базы данных в программные сущности (object relational mapping).

Также стоит отметить, что для избежания различных проблем, связанных с проектирование программного обеспечения, и для обеспечения гибкости в ходе реализации приложения была использована библиотека, осуществляющая внедрение зависимостей (dependency injection), — `Guice`. Также для борьбы с кончервативным кодом (boilerplate code) была использована библиотека `lombok`.

2.3 Методика и результаты тестирования

На этапе тестирования использовались библиотеки `junit` и `mockito`. Использование данных библиотек позволило реализовать тесты в соответствии с лучшими практиками. Отметим, что тестирование проводилось только для уровня бизнес логики. В силу большого количества тестов и проверяемых в них условиях в рамках данной работы не будет приводиться сколь-нибудь подробное описание тестовых методов. Вместо этого представим отчет по тестированию, сгенерированные при помощи утилиты `maven` и интегрированной среды разработки `IntelliJ IDEA`.

```
Running com.kspt.khandygo.em.services.AuthServiceTest
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.6 sec
Running com.kspt.khandygo.em.services.AwardsServiceTest
Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.274
↪ sec
Running com.kspt.khandygo.em.services.EmployeesServiceTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013
↪ sec
Running com.kspt.khandygo.em.services.OutOfOfficesServiceTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.062
↪ sec
Running com.kspt.khandygo.em.services.VocationsServiceTest
Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.053
↪ sec

Results :

Tests run: 34, Failures: 0, Errors: 0, Skipped: 0
```

Listing 2.1: Отчет по тестированию, сгенерированный в `maven`

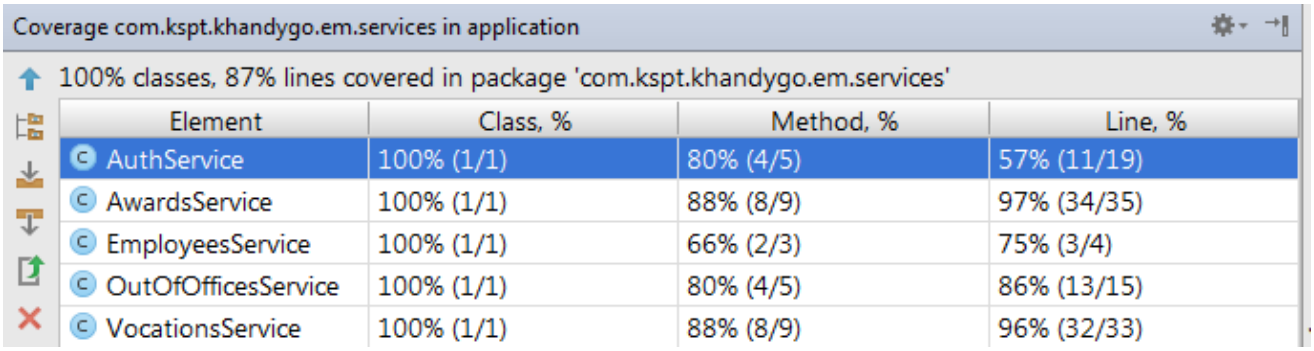

Рис. 7: Отчет по тестированию, сгенерированный в `IntelliJ IDEA`

2.4 Инструкция системного администратора

Для развертывания приложения необходимы следующие системные компоненты:

- Сервер `MySQL` версии 5.5 или выше.
- Утилита `maven` версии 3.1.0 или выше.
- Сервер приложений `wildfly` версии 10.1.0 или выше.

Теперь необходимо настроить `MySQL` сервер следующим образом:

- Порт 3307.
- Создать схему с именем `test`.

- Наполнить базу данных так, как показано в листинге 2.2.
- Создать пользователя с логином `em_admin` и паролем `1234`.
- Выделить пользователю `em_admin` права на чтение и запись в схеме `test`.

Никакой дополнительной настройки `maven` и `wildfly` не требуется. Для запуска сервера приложений в простейшем случае можно воспользоваться скриптом `standalone` (расширение `.bat` или `.sh` зависит от используемой операционной системы и средств), расположенным в директории `bin`. Теперь для размещения проекта на сервере необходимо выполнить команду `mvn clean install wildfly:deploy` из директории проекта. Здесь, конечно, предполагается, что на рабочей машине есть доступ к сети Интернет.

```
1  DROP SCHEMA IF EXISTS `test` ;
2  CREATE SCHEMA IF NOT EXISTS `test` DEFAULT CHARACTER SET utf8 ;
3  USE `test`;
4
5  CREATE TABLE `test`.`users` (
6    `id` INT NOT NULL AUTO_INCREMENT,
7    `login` VARCHAR(45) NOT NULL,
8    `password` CHAR(32) NOT NULL,
9    `name` VARCHAR(45) NOT NULL,
10   `manager_id` INT NULL,
11   `paymaster_id` INT NULL,
12   PRIMARY KEY (`id`));
13
14 ALTER TABLE `test`.`users`
15 ADD INDEX `fk_manager_id_idx` (`manager_id` ASC),
16 ADD INDEX `fk_paymaster_id_idx` (`paymaster_id` ASC);
17 ALTER TABLE `test`.`users`
18 ADD CONSTRAINT `fk_manager_id`
19   FOREIGN KEY (`manager_id`)
20   REFERENCES `test`.`users` (`id`)
21   ON DELETE NO ACTION
22   ON UPDATE NO ACTION,
23 ADD CONSTRAINT `fk_paymaster_id`
24   FOREIGN KEY (`paymaster_id`)
25   REFERENCES `test`.`users` (`id`)
26   ON DELETE NO ACTION
27   ON UPDATE NO ACTION;
28
29 CREATE TABLE `test`.`awards` (
30   `id` INT NOT NULL AUTO_INCREMENT,
31   `employee_id` INT NOT NULL,
32   `timestamp` BIGINT(64) NOT NULL,
33   `amount` BIGINT(64) NOT NULL,
34   `approved` TINYINT(1) NOT NULL DEFAULT 0,
```

```
35      `rejected` TINYINT(1) NOT NULL DEFAULT 0,
36      `cancelled` TINYINT(1) NOT NULL DEFAULT 0,
37      PRIMARY KEY (`id`));
38
39    ALTER TABLE `test`.`awards`
40    ADD INDEX `fk_awards_employee_id_idx` (`employee_id` ASC);
41    ALTER TABLE `test`.`awards`
42    ADD CONSTRAINT `fk_awards_employee_id`
43      FOREIGN KEY (`employee_id`)
44      REFERENCES `test`.`users` (`id`)
45      ON DELETE NO ACTION
46      ON UPDATE NO ACTION;
47
48    CREATE TABLE `test`.`out_of_offices` (
49      `id` INT NOT NULL AUTO_INCREMENT,
50      `employee_id` INT NOT NULL,
51      `timestamp` BIGINT(64) NOT NULL,
52      `duration` BIGINT(64) NOT NULL,
53      `reason` VARCHAR(256) NOT NULL,
54      `cancelled` TINYINT(1) NOT NULL,
55      PRIMARY KEY (`id`),
56      INDEX `fk_ooo_employee_id_idx` (`employee_id` ASC),
57      CONSTRAINT `fk_ooo_employee_id`
58        FOREIGN KEY (`employee_id`)
59        REFERENCES `test`.`users` (`id`)
60        ON DELETE NO ACTION
61        ON UPDATE NO ACTION);
62
63    CREATE TABLE `test`.`vocations` (
64      `id` INT NOT NULL AUTO_INCREMENT,
65      `employee_id` INT NOT NULL,
66      `timestamp` BIGINT(64) NOT NULL,
67      `duration` BIGINT(64) NOT NULL,
68      `approved` TINYINT(1) NOT NULL,
69      `rejected` TINYINT(1) NOT NULL,
70      `cancelled` TINYINT(1) NOT NULL,
71      PRIMARY KEY (`id`),
72      INDEX `fk_vocations_employee_id_idx` (`employee_id` ASC),
73      CONSTRAINT `fk_vocations_employee_id`
74        FOREIGN KEY (`employee_id`)
75        REFERENCES `test`.`users` (`id`)
76        ON DELETE NO ACTION
77        ON UPDATE NO ACTION);
```

Listing 2.2: Скрипт создания базы данных приложения

Для создания пользователя системы необходимо выполнить в `MySQL` команду вида

```
INSERT INTO users VALUES(null, login, MD5(pswd), name, m_id, p_id);
```

где: `login` — логин пользователя, `pswd` — пароль пользователя, `name` — имя пользователя, `m_id` и `p_id` — идентификационные номера менеджера и бухгалтера данного пользователя соответственно.

В случае правильной настройки всех компонент приложение должно быть доступно по адресу сервера приложений на порте с номером 8080 с добавлением `/application`.

2.5 Инструкция пользователя

Для того, чтобы начать пользоваться системой необходимо сначала пройти аутентификацию на заглавной странице. Для этого в соответствующие поля вводятся логин и пароль пользователя. Далее, в левой части экрана будет доступно меню.

- Для того, чтобы просмотреть подтвержденные заявки на отпуск и выплату премии необходимо перейти по пункту `Home`.
- Для того, чтобы просмотреть информацию о своем менеджере и бухгалтере, необходимо перейти по пункту `Stuff`.
- Для того, чтобы просмотреть информацию об отправленных, но еще не подтвержденных и не отклоненных заявках, а также заявках, направленных данному пользователю, необходимо перейти по вкладке `Proposals`.
- Для того, чтобы отправить заявку на предоставление отпуска необходимо нажать кнопку `Propose Vacation`.
- Для того, чтобы зарезервировать время отсутствия в офисе необходимо нажать кнопку `Reserve Out Of Office`.

3 Вывод

В рамках данной работы было разработано и реализовано сетевое приложение на языке Java с использованием технологии EJB. Считаю, что приложение разработано и реализовано на высоком уровне. Развитие приложения с функциональной точки зрения может быть продолжено в добавлении новых бизнес процессов. С точки зрения развертывания, в первую очередь необходимо добавить возможность конфигурации приложения из конфигурационного файла. В данном случае этого не было сделано по причине специфики работы выбранного сервера приложений с ресурсными файлами.

Дадим оценку приложения с точки зрения различных характеристик распределенных систем:

- **Прозрачность** обеспечивается методикой REST на уровне представления приложения.
- **Открытость** достигается тем, что приложение обладает сетевым интерфейсом.
- **Масштабируемость** приложения оценить достаточно сложно. Можно только предположить, что достаточно разместить приложение на распределенном сервере приложений.