Федеральное государственное автономное образовательное учреждение
высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий
Кафедра «Распределенные вычисления и компьютерные сети»

О Т Ч Е Т

Биллинговая система в сети оператора мобильной связи

по дисциплине
«Разработка корпоративных распределенных web-приложений
с помощью технологии Enterprise Java Beans»

Выполнил
студент гр. 63507/1 В.С. Чуприн

Руководитель
асс. И.В. Стручков

Санкт-Петербург
2017

Оглавление

Постановка задачи

Необходимо разработать корпоративное приложение для операторов биллинговой системы мобильной сети. Операторы сети могут регистрировать клиентов, создавать им счета, резервировать сессию разговора, списывая средства со счетов абонентов.

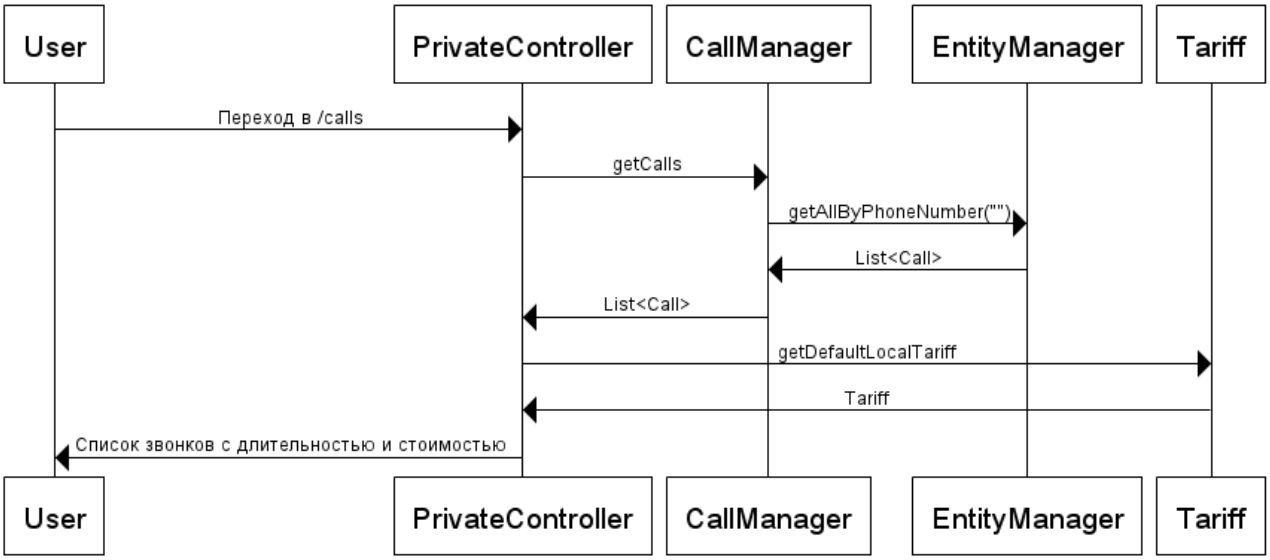
Счета абонентов хранятся в файле формата csv, информация о сессии разговора (номер телефона и продолжительность разговора) сериализуется и хранится в персистентном хранилище (реляционной базе данных в данной реализации).

Приложение разрабатывается с помощью технологии Enterprise Java Beans, поставляемой в Java Enterprise Edition. В приложении учитывается гибкость архитектуры для дальнейших разработок, воспроизводимое развертывание на сервере системного администратора и горизонтальное масштабирование вычислительных ресурсов.

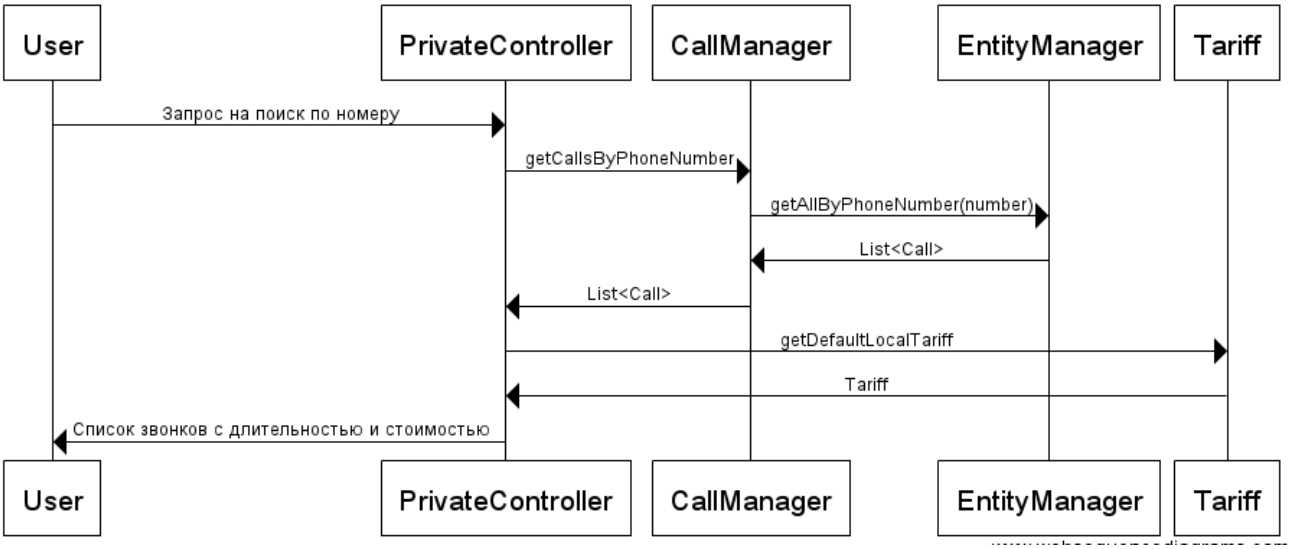
Варианты использования

Варианты использования представлены ниже в виде диаграммы последовательностей.

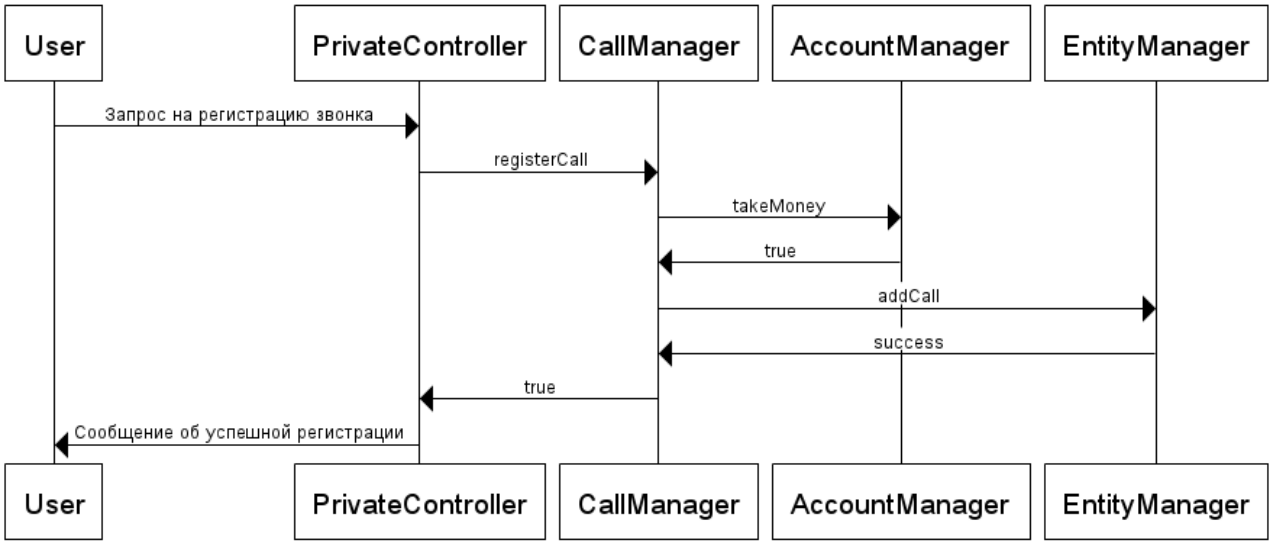
Отображение списка звонков



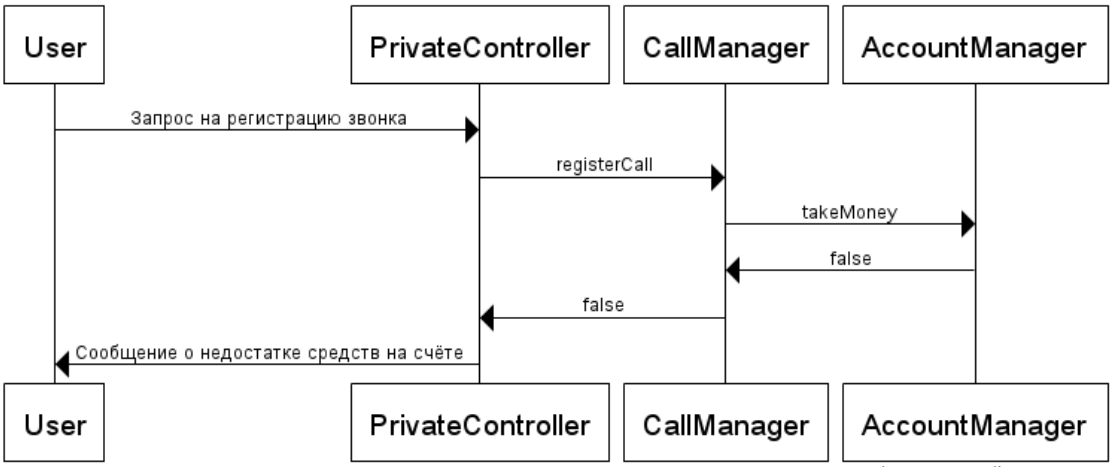
Поиск по номеру телефона



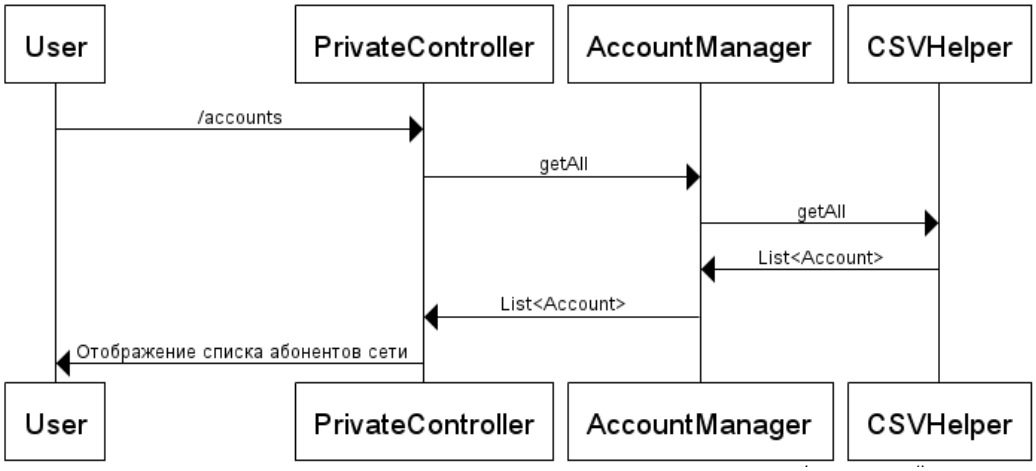
Регистрация звонка



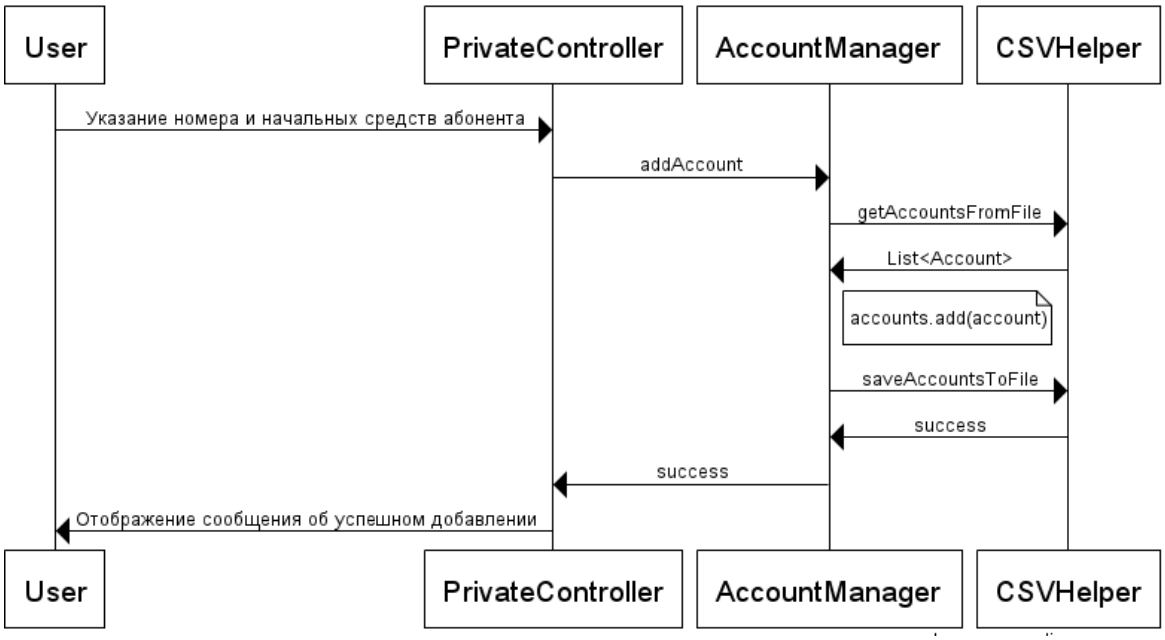
Регистрация звонка при недостатке средств



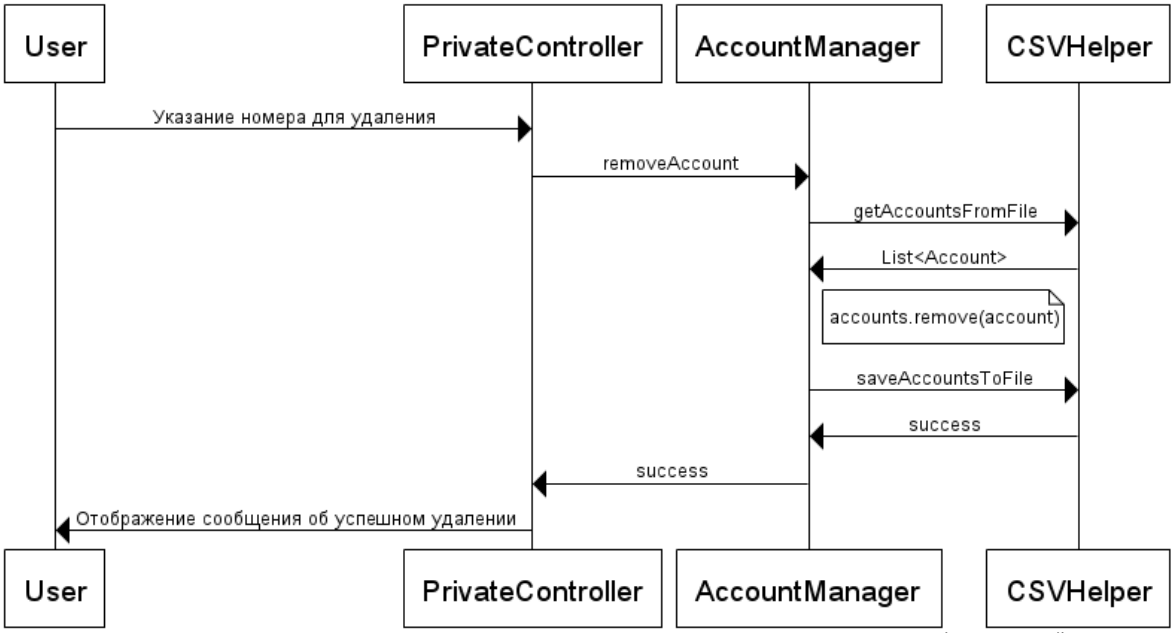
Получение списка абонентов сети



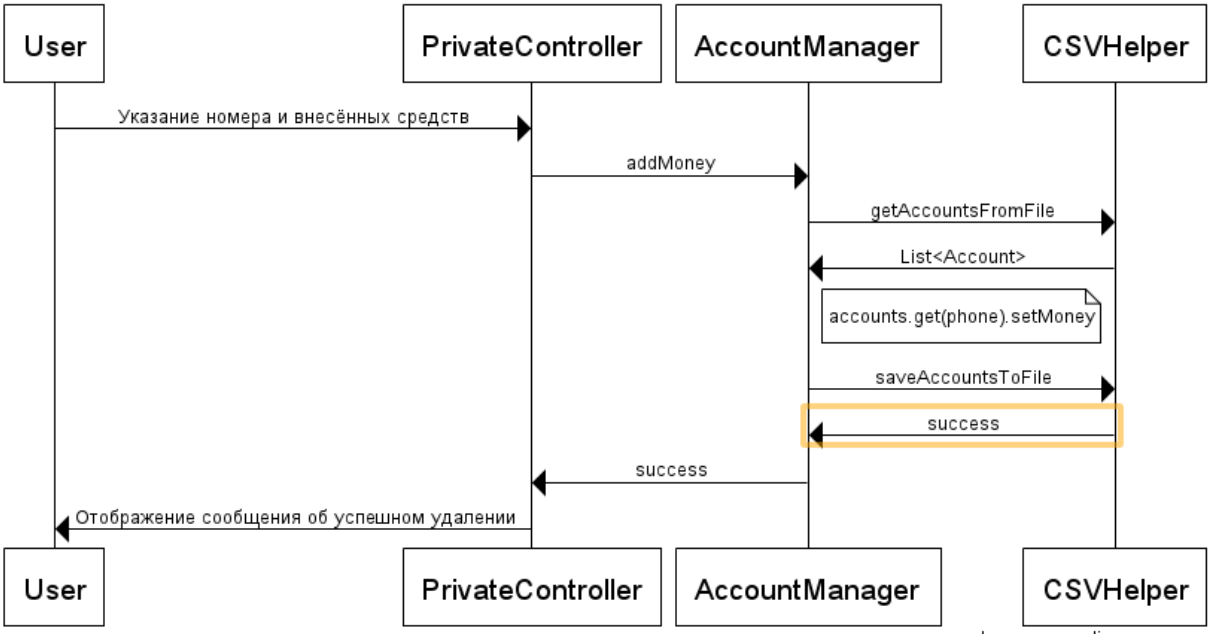
Регистрация абонента в сети



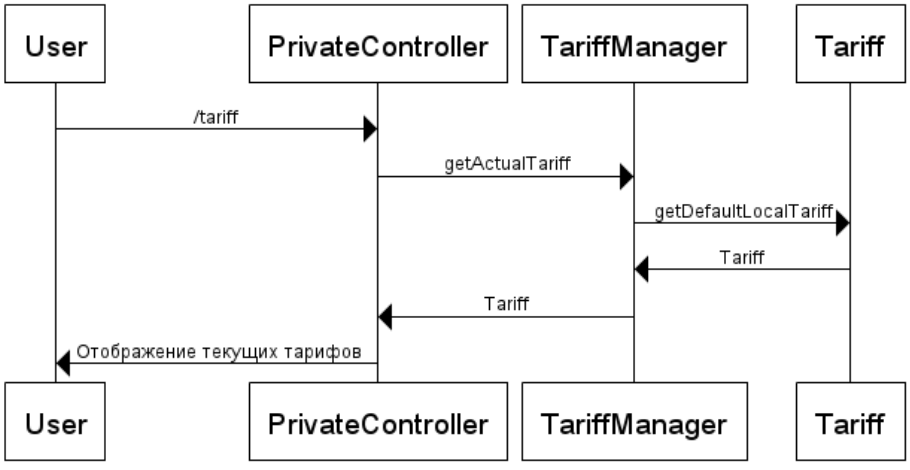
Удаление абонента из сети



Внесение средств на счёт абонента

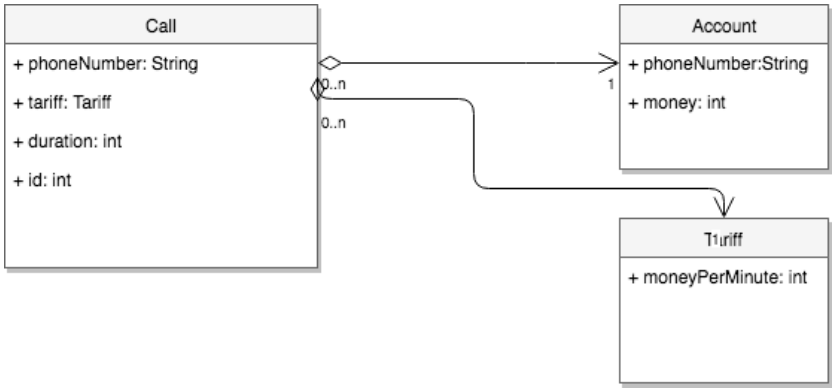


Получение списка тарифов



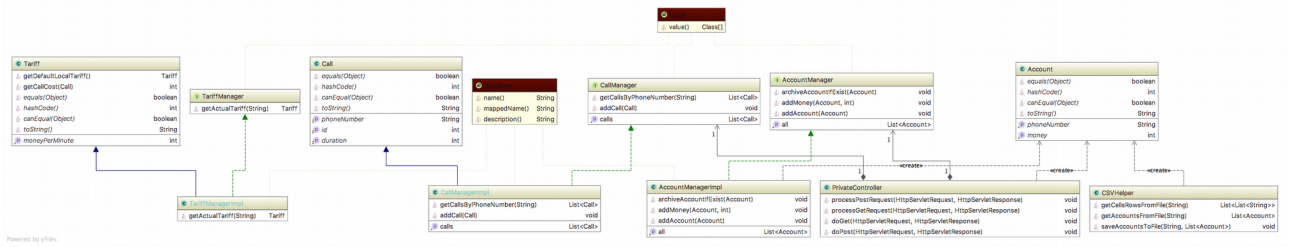
Модель предметной области

- Call – информация о сессии звонка.
- Tariff – тарифф с указанной поминутной стоимостью звонка.
- Account – информация об абоненте системы.



Объектно-ориентированное проектирование

Для реализации поставленной задачи была спроектирована следующая диаграмма.



Разработка программы

Как и сказано в задании – центральный элемент системы технология EJB и язык разработки Java. Сервер приложения – Glassfish 4. Было решено также сделать авторизацию пользователей (операторов), для чего использовались предоставляемая сервером система безопасности realm. Она предоставляет возможности для регистрации и авторизации пользователей на основе выбранной базы данных. Однако конфигурация такого решения была затруднительной и диагностика проблемы системой не предоставлялась (внутренние эксепшены записывались в лог, даже в банальных ошибках конфигурации). Так что была

выбрана admin-realm, которая записывает имена пароли и роли пользователей в файл, через консольную утилиту asadmin.

В качестве базы данных была выбрана PostgresQL, ввиду её простой установки на OSX и Linux. Также она используется и для тестирования, в то время как зачастую выбирают более легковесные базы данных. Такая гибкость выбора бд обеспечивается слоем JPA. Однако в моём случае PostgresQL на тестовом сервере не представляет никаких затрат, а одинаковая бд для тестов и релизного билда обеспечивает большую консистентность системы.

Для верстки был выбран bootstrap, таким образом обеспечивалась адаптивная верстка, предоставляющая адекватное размещение элементов для разных типов устройств.

Для тестирования используются junit и плагин в maven maven-surefire-plugin, который запускает по очереди все заданные тесты. Для тестов определена специальная папка с ресурсами (csv и sql файлы). В большинстве классов реализованно классическое unit-тестирование. При тестировании EJB имплементаций счёл уместным поднять реальный инстанс EntityManager, в первую очередь для себя: проще диагностировать упал glassfish из-за конфигурации базы данных (большинство случаев) или чего-то другого. Такие тесты правильнее назвать интеграционными. Далее для тестирования сервлетов зависимости были подменены с помощью Mockito.

Код размещался на github и к нему были настроена система для Continius Integration: Travis CI. В Travis была настроена база данных с заданным для разработки именем, именем пользователя, паролем. Чтобы развертывать актуальную схему к maven был подключен специальный плагин (sql-maven-plugin). Тот в свою очередь сносит (drop) и развертывает новую бд по заданным скриптам (схема таблиц и вставка данных). Также в github была интегрирована система для контроля покрытия кода тестами Coveralls. Чтобы создавать отчёты о покрытии тестами используется плагина для maven jacoco и плагин для отправки отчёта на свой аккаунт в coveralls. Таким образом в Travis запускается maven verify и далее указанные выше плагины.

Для развертывания готового приложения используется сервер Amazon Ec2 micro instance. Т.к. он является бесплатным в течении года и его небольших вычислительных мощностей хватает для данной курсовой работы. На нём также были развернуты glassfish и postgres.

Инструкция системного администратора

В данном разделена представлена инструкция для разворачивания приложения:

Шаг 1. Необходимое окружение:

- Java 8 или выше.
- Сервер GlassFish версии 4.1 и выше, подойдет и другой сервер приложений реализующий спецификации Java EE. Далее будет описана инструкция для сервера GlassFish;
- База данных Oracle Database 11c (скорее всего подойдет и гораздо более ранняя версия)

Шаг 2. Установка приложения:

- Накат схемы данных. Для наката схемы данных необходимо выполнить sql-скрипт create_tables.sql и fill_data.sql находящийся в папке main\java\resources\db\createbase.sql.
- Настройка соединения с базой данных:
  1. Добавьте в файл [domain_name]\config\domain.xml описание нового Connection pool:
     <resources>

     <jdbc-connection-pool datasource-classname="org.postgresql.ds.PGSimpleDataSource" name="postgres_pool" res-type="javax.sql.DataSource">
     <property name="connectionAttributes" value=";create=true"></property>
     <property name="user" value="root"></property>
     <property name="password" value="root"></property>
     <property name="portNumber" value="5432"></property>
     <property name="databaseName" value="postgres"></property>
     <property name="serverName" value="localhost"></property>
     <property name="driverType" value="thin"></property>
     </jdbc-connection-pool>
     <jdbc-resource pool-name="postgres_pool" jndi-name="jdbc/postgres_pool"></jdbc-resource>
     </resources>
  2. Добавить в файл [domain_name]\config\domain.xml для настройки sequrity realm:
     <security-service activate-default-principal-to-role-mapping="true">
     <auth-realm classname="com.sun.enterprise.security.auth.realm.file.FileRealm" name="admin-realm">
     <property name="file" value="${com.sun.aas.instanceRoot}/config/admin-keyfile"></property>
     <property name="jaas-context" value="fileRealm"></property>
     </auth-realm>
     </security-service>
- Деплой war-архива приложения на сервер glass-fish. Приложение будет доступно по адресу http://[host-name]:[glassfish_port]/ewa-app
- В консоли добавить всех пользователей asadmin create-file-user --groups admin username и далее ввести свой пароль и пароль будущего пользователя.

Инструкция пользователя

- Для начала работы с приложением необходимо перейти по адресу http://[host-name]:[glassfish_port]/ewa-app

Выводы

Были изучены основы разработки программного обеспечение с использованием объектно-ориентированного проектирования на основе технологии Enterprise Java Beans.

В качестве дальнейших расширений системы можно рассматривать:

- Повышение покрытия кода тестами
- Выделение бд для тестирования для ускорения разработки

- Интеграция утилит для отслеживания code style и статического анализа
- Регистрация пользователей и хранение аккаунтов в бд

Список внешних ресурсов

1. https://github.com/vlastachu/billing_operator репозиторий с текущим проектом.
2. https://travis-ci.org/vlastachu/billing_operator текущий статус сборки проекта.
3. https://coveralls.io/github/vlastachu/billing_operator текущий статус покрытия кода тестами.
4. http://ec2-52-39-144-186.us-west-2.compute.amazonaws.com:8080/ewa-app/ сервер с запущенным приложением (актуально на краткосрочный период).