МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет компьютерных наук

Отчет по практикуму «Технологии программирования»

Проектная документация к задаче «Веб маркет»

Выполнили:  
Шачнева А.И., Астрелин М. А., Семенко А. А.

Воронеж 2019

[Введение 2](#_Toc7991686)

[Постановка задачи 2](#_Toc7991687)

[Анализ предметной области 3](#_Toc7991688)

[Описание структуры проекта с помощью UML-диаграмм 3](#_Toc7991689)

[ Диаграмма классов 3](#_Toc7991690)

[ Диаграмма последовательностей 7](#_Toc7991691)

Введение

Интернет магазины, являющиеся по совместительству площадками для частных магазинов, популярны в России. Например, каждому знаком AliExpress, но кроме него существуют и другие онлайн-магазины. Пользователю не удобно мониторить интересующие товары сразу на нескольких площадках, ведь приходится регистрироваться сразу на нескольких подобных сервисах. Хорошим решением этой проблемы было бы написание агрегатора товаров со всех этих магазинов – это бы упростило пользователям поиск товаров.

Постановка задачи

Цели:

Создать сервис, позволяющий искать товары из других сервисов через свой интерфейс

Создать систему, позволяющую легко добавить новые магазины в приложение

Целевая аудитория:

Люди, желающие облегчить себе поиск товаров из нескольких магазинов

Требования:

Регистрация и вход в систему

Поиск по всем добавленным в систему магазинам

Возможность добавления товаров в избранное

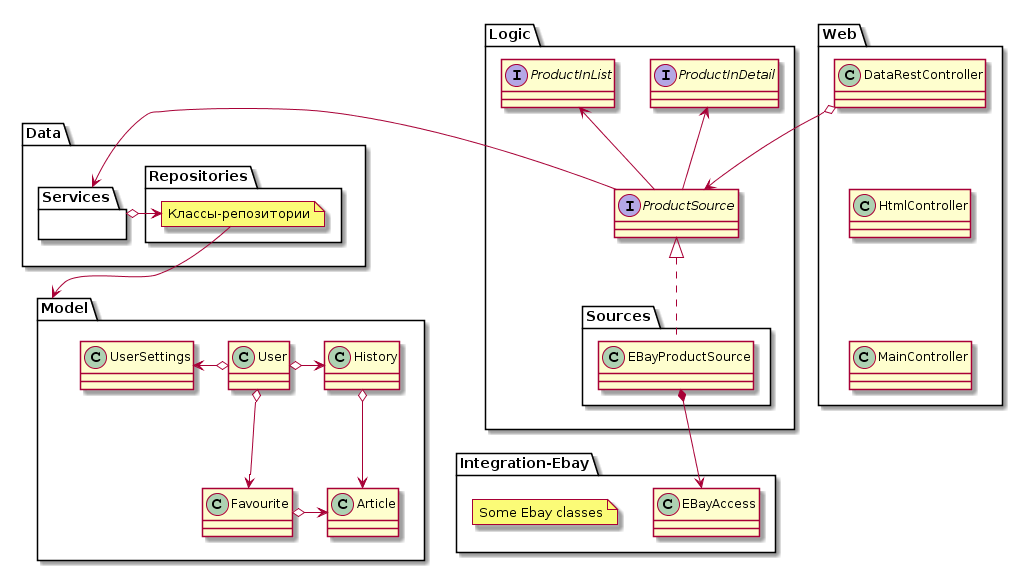
Возможность просмотреть историю просмотров

Анализ предметной области

Был проведен анализ возможных аналогов приложения. Среди возможных конкурентов существует Яндекс.Маркет, но принцип его работы отличается от работы этого проекта. В Яндекс.Маркете магазины регистрируют себе бизнес-аккаунты и связывают собственные базы данных товаров с API Яндекса. Такой подход требует от приложения быть популярным, чтобы к нему проявили интерес крупные магазины. Приложение «Веб маркет» использует публичное API прочих онлайн-магазинов для выполнения выборки по их товарам. Подходящее API есть у онлайн-магазина eBay.

Описание структуры проекта с помощью UML-диаграмм

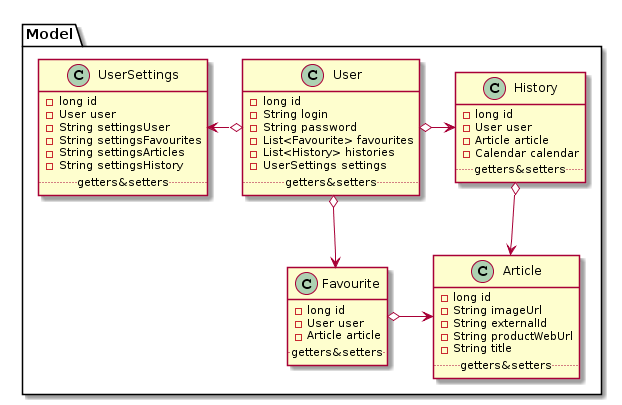
Диаграмма классов

Диаграмма классов серверной части приложения: 

На диаграмме можно видеть, что приложение состоит из четырёх основных модулей – модель, данные, логика и веб. Есть ещё побочные модули (Integration-Ebay), который использует модуль логики – это модули взаимодействия с прочими сервисами онлайн-магазинов. Они используются только в модуле логики.

Далее - серверная часть более подробно.

Модуль “Модель”:

**

В этом модуле содержатся классы модели. Пояснения к классам:

Article – позиция в магазине. Все страницы из магазинов не сохраняются в БД, но если какая-то позиция была добавлена в избранное или в историю просмотров, то она сохраняется. Содержит в себе необходимые для корректного отображения представления о товаре – ссылку на товар в магазине, изображение товара, идентификатор товара в магазине и его заголовок.

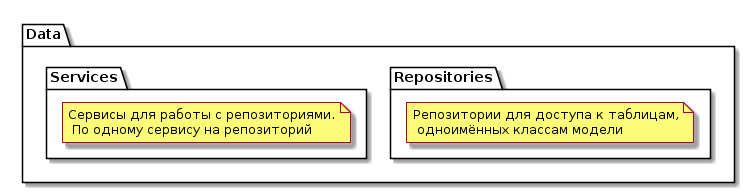
History – запись о просмотре товара пользователем. Имеет связь с пользователем, который товар посмотрел, и с товаром, который он посмотрел, а также время просмотра.

Favorite – запись об избранном товаре. Содержит ссылку на пользователя и на товар. Очередность добавления в избранное можно определить по id.

User – пользователь. Содержит списки избранных и просмотренных товаров, данные авторизации и ссылку на пользовательские настройки.

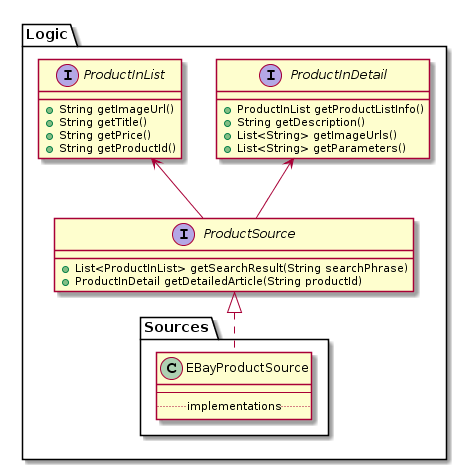
UserSettings – класс с информацией о настройках пользователя. Имеют вольный формат (text), могут быть null. По умолчанию не заданы. Этот класс может быть использован при дальнейшем расширении проекта.

*Модуль «Доступ к данным»:*

**

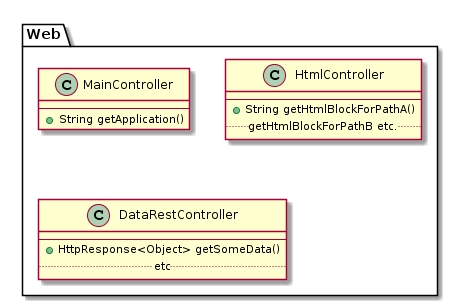
Этот класс используется для доступа к базе данных. Этот модуль использует фреймворки Hibernate и Spring Data JPA. Они вместе позволяют автоматически генерировать SQL-запросы к базе данных, а также реализовывать методы для рутинных операций вроде добавления новых кортежей или выборки. Реализация генерируется в репозиториях. Сервисы предоставляют репозитории и являются дополнительной оболочкой над репозиториями. Позволяют агрегировать запросы к репозиториям.

*Модуль “Логика”:*

**

Модуль логики позволяет делать запросы к прочим онлайн-магазинам. Каждый модуль интеграции с онлайн-магазином написан отдельно, изолированно от проекта. А в проекте (в данном случае в пакете Sources) имеются адаптеры для всех модулей.

Модуль “Веб”:



Этот модуль обрабатывает запросы от клиента. Делится на три класса-контроллера:

Main – контроллер, возвращающий приложение пользователю.

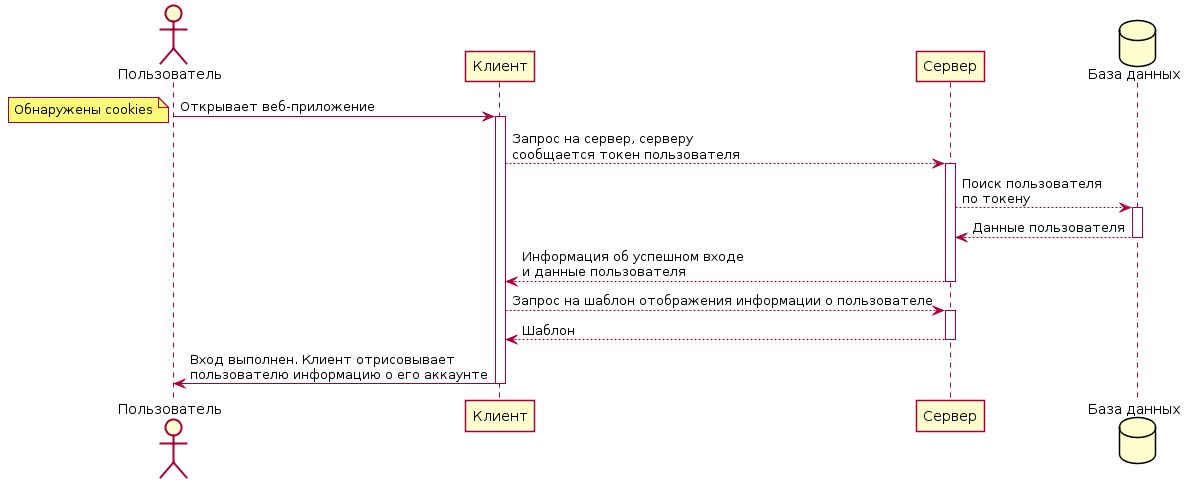
Html – контроллер, возвращающий пользователю некоторые шаблоны приложения. Это позволяет пользователю начать пользоваться приложением, не загружая некоторой части html-шаблонов, ускоряя его первый запуск. В дальнейшем он по запросу получает шаблоны, которые у себя сохраняет.

DataRest – контроллер, отвечающий клиенту данными в JSON-формате.

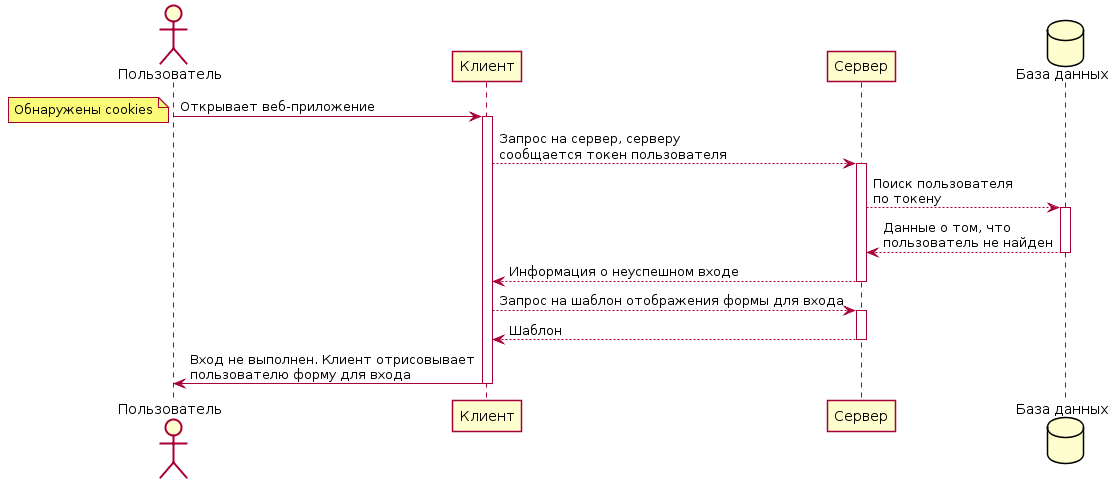
Модуль использует фреймворк Spring MVC для реализации контроллеров. Для сборки страниц используется фреймфорк Apache Tiles 3. Он позволяет собирать html-страницы на стороне сервера, а потом возвращать их клиенту. Но собирается только первая страница, потом веб-приложение работает как SPA, получая данные из DataRestContoller и шаблоны из HtmlController.

Диаграмма последовательностей

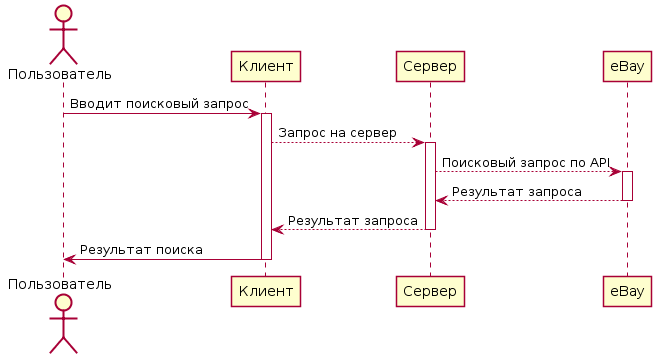
Эта диаграмма отображает процесс входа пользователя на сайт, если у него обнаружены cookies:



Эта диаграмма отображает процесс входа пользователя на сайт, если у него обнаружены cookies, не являющиеся актуальными. В случае, если cookies не будут найдены, запрос на сервер с токеном пользователя будет опущен, остальное то же самое:



Поисковый запрос происходит по следующему сценарию:



Для примера было показано взаимодействия с eBay. С другими сервисами будет реализовано подобное взаимодействие. Параллельно выполняются такие запросы для всех магазинов. Результаты потом сортируются и отображаются пользователю.