**Задание**

Сервис по поиску товаров среди онлайн-магазинов.

**Роли**

В системе существует 2 роли:

* Пользователь - клиент, который хочет сделать выборку товаров по онлайн-магазинам;
* Администратор - человек, управляющий содержимым сервиса.

**Функциональные требования**

Для пользователя:

1. Регистрация и авторизация
2. Выбор используемых при поиске онлайн-магазинов
3. Поиск товаров по названию
4. Расширенный поиск товаров по атрибутам
5. Сортировка и фильтрация полученного результата
6. Импорт и экспорт списка товаров в XML
7. Управление избранным - добавление и удаление товаров
8. Выставление оценки товарам
9. Выбор отображаемой валюты

Для администратора:

1. Возможности обычного пользователя
2. Управление списком доступных онлайн-магазинов

a. Добавление нового магазина

b. Удаление существующего магазина

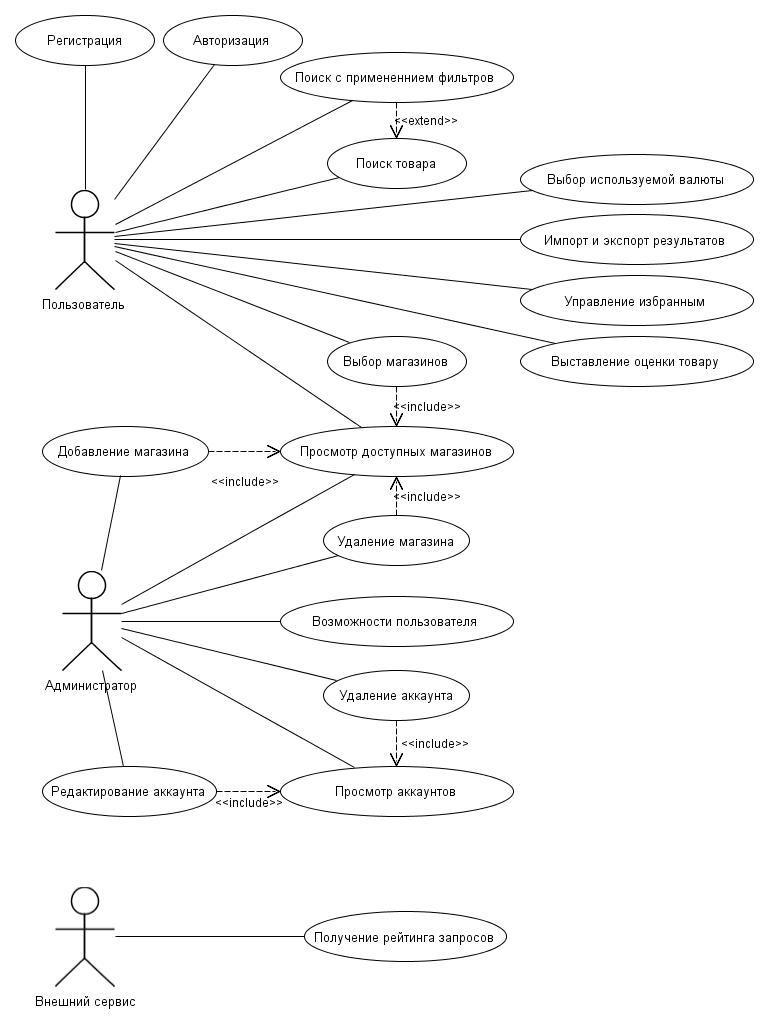
1. Управление пользователями

a. Удаление пользователя

b. Редактирование данных пользователя

1. Обновление базы товаров

**Варианты использования**



**Текстовое описание вариантов использования**

Варианты использования для пользователя:

1. Регистрация

1.1. Пользователь вводит логин и пароль.

1.2. После проверки данных регистрация успешно завершается.

1.3. Альтернатива: пользователь с таким логином уже существует в системе или введен недопустимый пароль (менее трех символов). Предлагается ввести другой логин или пароль.

1. Авторизация

2.1. Пользователь вводит логин и пароль.

2.2. Если данные проходят проверку, авторизация успешно завершается.

2.3. Альтернатива: данные не проходят проверку. Предлагается заново ввести логин и пароль.

1. Поиск товара

3.1. Пользователь задает необходимые условия поиска (наименование **товара**) и, по желанию, дополнительные **свойства**: категория, производитель, год производства, цена. Для даты и цены возможно задать условия в виде диапазона.

3.2. Выводится результат поиска – список товаров, подходящих по заданным условиям. Записывается **история** поиска.

3.3. Пользователь может отсортировать результаты поиска по названию, стоимости или рейтингу товара.

3.4. Для любого товара в результатах поиска имеется возможность просмотреть подробную информацию о товаре, перейти на страницу в Интернете с описанием товара, добавить товар в избранное и поставить ему оценку с помощью соответствующих кнопок или команд меню.

3.5. Список товаров в любой момент можно экспортировать в XML, а также можно загрузить список товаров из XML.

1. Управление профилем

4.1. Пользователь может просмотреть список **магазинов** и отметить те, которые он хочет использовать для поиска.

4.2. Пользователь может просмотреть список товаров, которые он добавил в избранное.

Варианты использования для администратора:

1. Регистрация, поиск товаров и управление профилем – аналогично пользователю.
2. Управление магазинами

2.1. Администратор просматривает список магазинов.

2.2. Имеется возможность добавить в список новый магазин, при этом администратор заполняет необходимую информацию о магазине.

2.3. Имеется возможность удалить существующий магазин.

1. Управление пользователями

3.1. Администратор просматривает список аккаунтов в системе.

3.2. Имеется возможность редактировать данные любого пользователя.

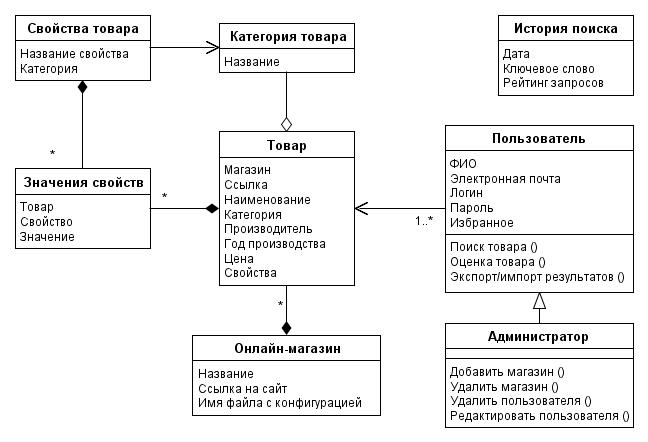
3.3. Имеется возможность удалить любого пользователя.

1. Администратор может запустить функцию обновления базы данных товаров из доступных онлайн-магазинов

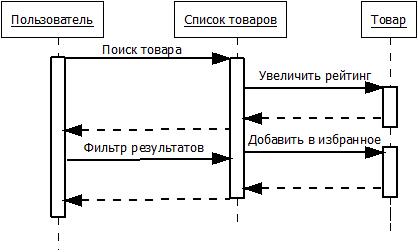
Варианты использования для внешнего сервиса:

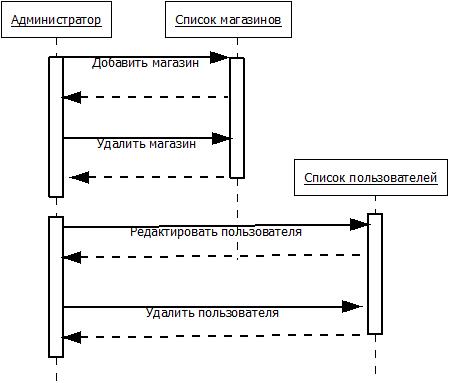
1. Сервис запрашивает наиболее употребляемые ключевые слова из запросов пользователей, указывая количество слов, которое он хочет получить.
2. Возвращается XML с запрошенным количеством наиболее употребляемых ключевых слов.
3. Альтернатива: если ключевых слов в базе меньше, чем требуется в запросе, высылаются все ключевые слова.

**Разработка статической объектной модели предметной области**



**Разработка динамической модели**





**Проектирование слоя бизнес-логики**

В качестве паттерна для реализации бизнес-логики был выбран паттерн Domain Model (модель предметной области). Данный паттерн очень хорошо подходит для отображения объектно-ориентированной модели, однако требует наличия опыта работы с ООП.

Бизнес-логика приложения реализована в пакетах logic и utils и включает следующие классы:

* Product
* User
* Administrator
* Store
* Category
* HistoryEntry
* HashManager
* StringComparator
* UserUtils
* ProductUtils

**Проектирование слоя источников данных**

В качестве паттерна для реализации данного слоя было решено выбрать паттерн Data Mapper. Данный шаблон хорошо сочетается с паттерном Domain Model и позволяет абстрагировать приложение от конкретной реализации слоя доступа к данным. Из особенностей реализации можно отметить введение дополнительного поля для пользователя, которое указывает на его тип. Это позволило эффективно отобразить наследование класса Administrator от класса User, поля которых полностью совпадают.

Слой источников данных реализован в пакете db и включает в себя следующие классы:

* AbstractMapper
* CategoryMapper
* ConnectionManager
* HistoryEntryMapper
* ProductFilter
* ProductMapper
* UserMapper
* StoreMapper

**Проектирование сервисного слоя**

Сервисный слой инкапсулирует логику приложения, предоставляя набор доступных операций. Также в сервисном слое находятся классы для работы с внешними сервисами. Данный слой реализован в пакетах service и webservice и включает в себя следующие классы:

* CategoryService
* HistoryService
* ProductService
* StoreService
* UserService
* XMLBuilder
* XMLReader
* HTTPClient
* TopQueriesServer
* TopQueriesHandler

**Проектирование слоя представления**

Реализация графического интерфейса с использованием библиотеки Swing находится в пакете ui. Основная информация - результат поиска, - отображается с помощью таблицы. Для большей эффективности были разработаны классы модели таблицы и рендереры для каждого столбца. Они расположены в пакете ui.resultsTable. Сервисный слой включает в себя следующие классы:

* ui.AddStoreJDialog
* ui.AuthorizationJDialog
* ui.EditUserJDialog
* ui.FindGoodsJFrame
* ui.PreferencesJDialog
* ui.ProductInfoJDialog
* ui.ProductInfoTableModel
* ui.RegistrationJDialog
* ui.StoresJPanel
* ui.UsersJPanel
* ui.UsersTableModel
* ui.resultsTable.ButtonCellRenderer
* ui.resultsTable.ButtonMouseListener
* ui.resultsTable.CurrencyCellRenderer
* ui.resultsTable.ImageCellRenderer
* ui.resultsTable.JEditorPaneCellRenderer
* ui.resultsTable.NumberCellRenderer
* ui.resultsTable.ResultsTableModel

**Тестирование приложения**

Тестирование проводилось вручную и автоматически, при помощи модульных тестов. Ручное тестирование использовалось для тестирования графического интерфейса. Все unit-тесты содержатся в пакете test, в котором вложенные пакеты отображают пакеты с исходным кодом в проекте.

**Вывод**

В результате работы были изучены основные архитектурные шаблоны проектирования, и часть из них применена на практике. Были рассмотрены типовые решения организации различных слоев приложения: бизнес-логики, слоя доступа к данным, сервисного слоя. В результате был реализован рабочий проект в соответствии с исходными требованиями. Можно отметить, что использование паттернов проектирования весьма эффективно ввиду того, что позволяет более просто и гибко писать код и повышает его читабельность. Однако их использование должно быть целесообразным, чтобы не допускать чрезмерного усложнения кода.