**Формулировка проблемы**

Программисты, врачи и все, кто умеют работать мозгами и руками часто приходится сталкиваться с такой ситуацией, как отсутствие умения приготовить еду. С одной стороны, это забавный стереотип, а с другой, всё это не так уж и далеко от правды. Часто людям приходится сталкиваться с французским меню «Шаро'м покати'» в своём холодильнике. Это когда кажется, что еды совсем нет, а кушать хочется, но это только с первого взгляда. Удивительно, что из таких ингредиентов как яиц, петрушки, майонеза и куска хлеба можно приготовить большое количество блюд.

Современные технологии призваны делать нашу жизнь легче и удобнее. Трудно сказать, чего от них больше: пользы или вреда. Всё зависит от того, как их использовать. Практическая часть данной дипломной работы помогает обычному пользователю на кухне, не зависимо от того, новичок он или профессионал.

**3.1. Процессы автоматизации помогающие на кухне**

Кажется, что веб приложение мало чем может помочь на кухне, ведь на кухне нужно только умение готовить. Все, кто так думают, глубоко заблуждаются. Из имеющихся в холодильнике продуктов, нужно подобрать правильный рецепт из «бабушкиного» кулинарного блокнота. А в случае, если в холодильнике окажется замороженный осьминог или устрицы, блокнот будет бесполезен. Практическая часть данной дипломной работы реализует веб приложение, автоматизирующее такие процессы как:

* Подбор рецептов блюд из ингредиентов, хранящихся в холодильнике пользователя.
* Случайный подбор блюд из ранее добавленных в «избранное».
* Упрощение управления и учёта продуктов в холодильнике.
* Добавление авторского блюда для всех пользователей приложения.

**3.2. Анализ рынка конкурентов**

**3.2.1. Мобильное приложение: «Что готовим?»**



*Рисунок 3.1. Интерфейс мобильного приложения «Что готовим ?»*

**Описание приложения:**

Основная функция приложения - подбор рецептов блюд, которые можно приготовить прямо сейчас. Перенеся список содержимого холодильника в приложение остаётся только выбирать блюда из предложенного списка.

Приложение имеет рецептурную базу для более чем 850 блюд и будет регулярно дополняться новыми рецептами.

Каталог рецептов содержит категории:

* Супы
* Салаты
* Закуски
* Горячие блюда
* Гарниры
* Каши
* Вегетарианские блюда
* Выпечка
* Десерты
* Рецепты для мультиварки
* Соусы
* Напитки

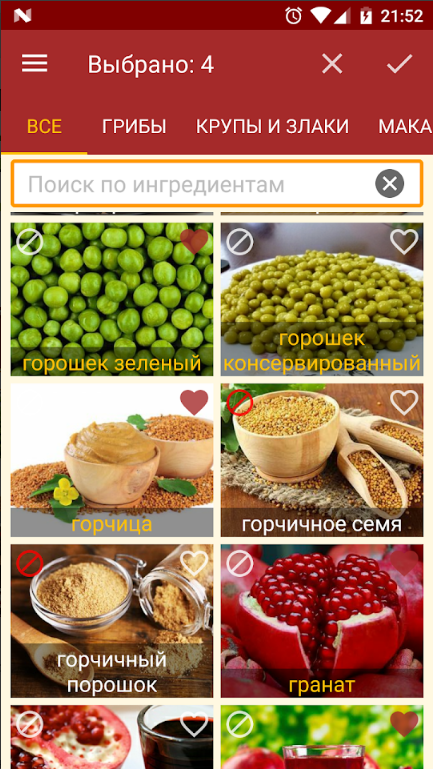
**Основные преимущества приложения:**

* Приложение составляет блюда из имеющихся продуктов в холодильнике, что очень удобно.
* Рецепты разнообразные на любой вкус и разной ценовой категории.
* Приложению не нужен доступ в интернет.

**Основные недостатки приложения:**

* Обновление списка всех блюд выходит только с обновление приложением.
* Приложение только для мобильной платформы и не имеет веб сервиса.
* Неудобный интерфейс. Выбирать продукты, слева галочка, название зачеркивается, справа звездочка - что всё это значит, непонятно. Рецепты блюд читать неудобно - сплошной текст, без абзацев и знаков препинания.
* Медленно работает приложение. Видно задержки при переходах между вкладками.
* Неправильно работает поиск блюд из имеющихся продуктов - рецепты предлагаются из других продуктов.

**3.2.2. Мобильное приложение «Подбери рецепт»**



*Рисунок 3.2. Интерфейс мобильного приложения «Подбери рецепт?»*

**Описание приложения:**

В приложении находятся простые рецепты и блюда разных стран мира с фотографиями. Все рецепты доступны без интернета. Основная функция приложения - подбор блюд по имеющимся дома продуктам.

Найденные рецепты отсортированы по наибольшему количеству совпадающих продуктов, а также по категориям блюд. Найденные ингредиенты выделяются цветом внутри рецепта.

Каждый рецепт содержит фотографию, список ингредиентов, калорийность, соотношение БЖУ и подробное описание.

В приложении встречаются рецепты из следующих категорий:

блины и оладьи, блюда для мультиварки, блюда на завтрак, вторые блюда, гарниры, десерты, заготовки, закуски, здоровое питание, каши, напитки, несладкая выпечка, рыба и морепродукты, салаты, сладкая выпечка, соусы, супы.

**Основные преимущества приложения:**

* Быстро работает, без задержек.
* Есть веб-сервис и возможность использовать приложение офлайн.
* Удобное меню и управление.
* Большой список рецептов.
* Поиск блюд из выбранных продуктов работает корректно.

**Основные недостатки приложения:**

* Слишком мало информации о процессе готовки.
* Отсутствие авторских блюд. Пользователи не могут выкладывать в приложение свои блюда.
* Отсутствие веб приложения. Нет возможности пользоваться персональным компьютером или ноутбуком.
* Отсутствие фильтра всех рецептов.
* Устаревший дизайн.
* Постоянная реклама внизу экрана.

**3.3. Основной функционал практической части дипломной работы**

После проведённого исследования, на базе выявленных преимуществ и недостатков вышеописанных приложений, в данной дипломной работе было принято решение разработать такой основной функционал для приложения, как поиск блюд по имеющимся продуктам в холодильнике. Поиск происходит по большому количеству рецептов. Список рецептов пополняется не только стандартными, но и авторскими блюдами, которые могут добавить обычные пользователи. Найденные блюда будут отсортированы по наибольшему количеству совпадающих блюд, указанных в холодильнике. Холодильник может использоваться несколькими пользователями, что будет удобно для всей семьи.

Рецепты блюд, в свою очередь, подробно расписаны. Для каждого этапа приготовления блюд прилагаются фотографии. Интересным рецептам можно поставить «лайк» или прокомментировать.

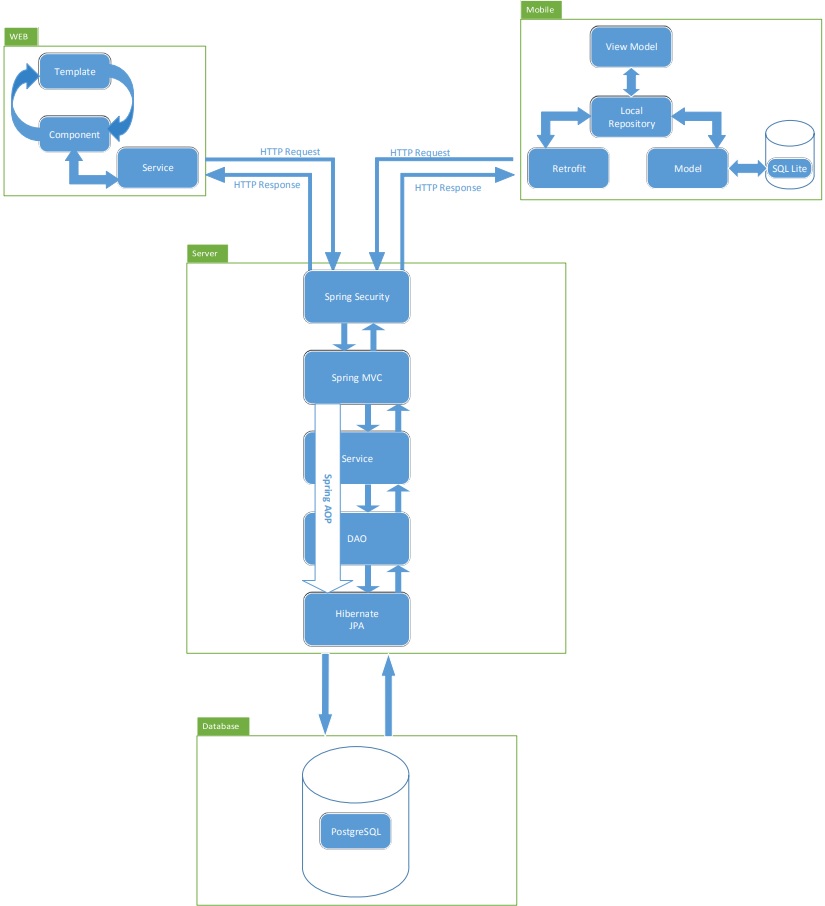
Добавление блюд в избранное позволит быстрее получать доступ к полюбившимся рецептам. Для облегчения выбора, из списка избранных блюд можно подобрать случайное блюдо. Веб-приложение имеет сайт и мобильное приложение, которые имеют доступ к общим данным т.е. изменения на сайте будут отображены в мобильном приложении и наоборот. Мобильному приложению не обязательно нужен интернет. В случае, если интернет отсутствует, под рукой всегда будут любимые рецепты блюд.

Список основных функций приложения:

* Поиск блюд по имеющимся продуктам в холодильнике.
* Возможность нескольким пользователям использовать один общий холодильник.
* Возможность сохранения любимых рецептов в "Избранном", чтобы иметь к ним быстрый доступ.
* Множество категорий позволяет сразу перейти к интересующему блюду.
* Подробное пошаговое описание рецепта с фотографиями.
* Возможность добавлять авторские рецепты для всех пользователей.
* Возможность ставить «лайки» и комментировать блюда.

**3.4. Описание разработанного приложения**

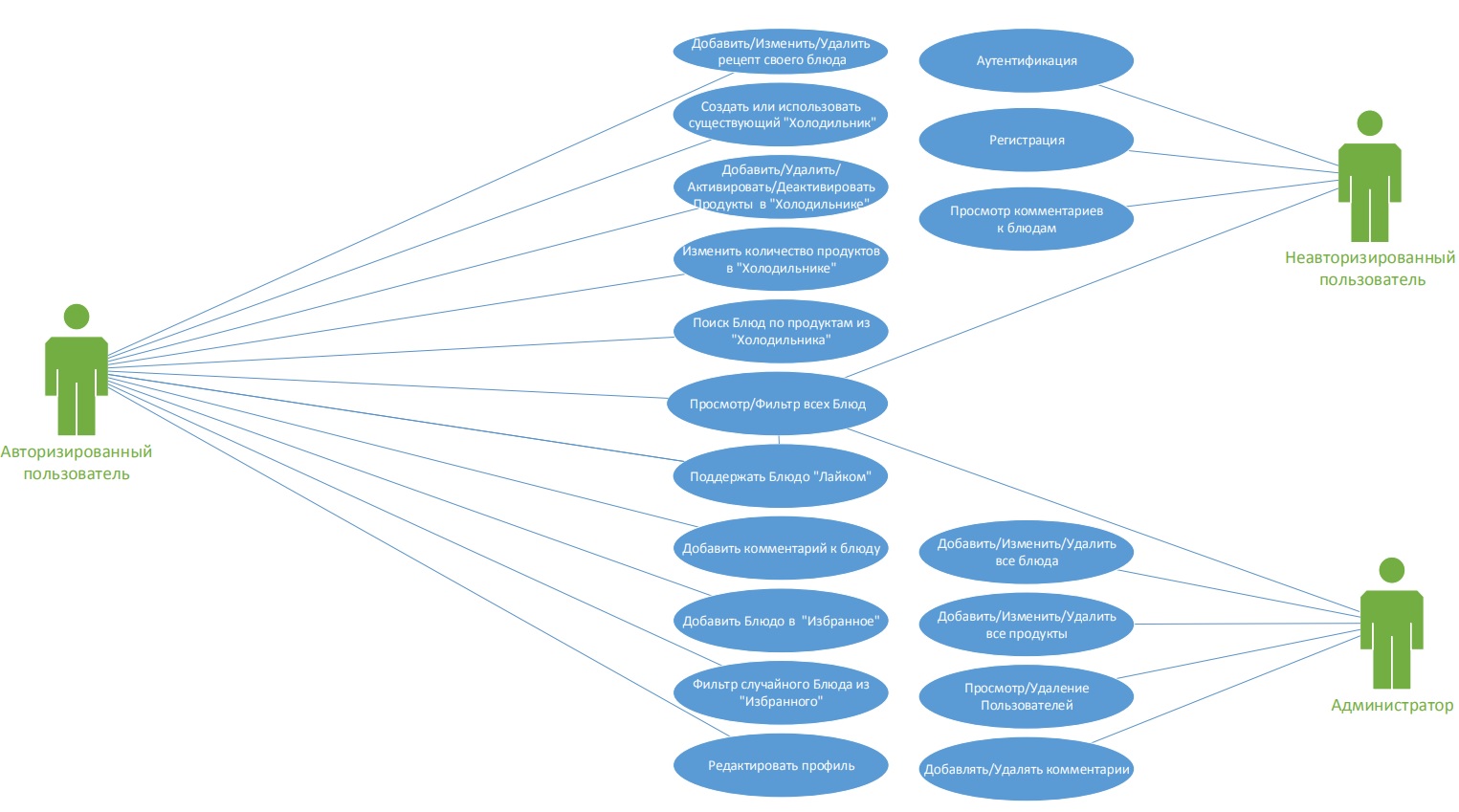
**3.4.1. Компонент диаграмма (Component diagram)**



*Рисунок 3.3. Компонент диаграмма веб-приложения.*

* **Web**
  + Template - шаблон, который указывает, как будет выглядеть компонент.
  + Component - основные строительные блоки приложения Angular.
  + Service - запрашивает данные у сервера и предоставляет их, инкапсулирует бизнес-логику, различные вычислительные задачи.
* **Mobile**
* View model – контролирует жизненный цикл отображения. Предоставляет данные из local repository.
* Local Repository – запрашивает данные у REST API сайта, принимает и обрабатывает постоянные данные из model.
* Retrofit – библиотека, которая упрощает взаимодействие с REST API сайта.
* Model – Модель предоставляет данные и методы работы с ними: запросы в базу данных, проверка на корректность.
* SQL Lite – компактная встраиваемая СУБД.
* **Server**
* Spring Security - предоставляет механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности.
* Spring MVC - обеспечивает архитектуру паттерна Model — View — Controller при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними.
* Service – место выполнения бизнес-логики используя DAO классы.
* DAO – (Data Access Object) облегчает работу с такими технологиями доступа к данным, как Hibernate, JPA.
* Hibernate, JPA – предназначено для объектно-реляционного отображения (ORM). Связывает Java классы с таблицами базы данных, но также предоставляет средства для автоматического построения запросов и извлечения данных.
* **Database**
* PostgreSQL - свободная объектно-реляционная система управления базами данных.

**3.4.2. Диаграмма прецедентов использования (Use case diagram)**



*Рисунок 3.4. Диаграмма прецедентов использования веб-приложения.*

В диаграмме присутствуют 3 вида пользователей :

* **Неавторизированный пользователь** – пользователь, который не вошёл в систему.
* **Авторизированный пользователь** – пользователь, который вошёл в систему.
* **Администратор** – пользователь, который вошёл в систему и имеет доступ к управлению системой.

**Действия авторизированного пользователя в системе:**

* *Добавить/Изменить/Удалить рецепт своего блюда –* пользователь имеет право на добавление блюда в систему и дальнейшее его изменение и удаление.
* *Создать или использовать существующий "Холодильник" –* пользователь имеет право на создание «Холодильника» (хранилища продуктов) или использовать существующий «Холодильник» т.е. использовать хранилище продуктов другого пользователя с разрешения основателя хранилища продуктов.
* *Добавить/Удалить/Активировать/Деактивировать Продукты в "Холодильнике" –* пользователь имеет право на добавление, удаление, отмечать продукт активным или неактивным в «Холодильнике».
* *Изменить количество продуктов в "Холодильнике" -* пользователь имеет право на изменение количества продуктов уже добавленных в «Холодильник».
* *Поиск Блюд по продуктам из "Холодильника" -* пользователь имеет право на поиск блюд в ингредиенты, которых входят активные продукты из «Холодильника».
* *Поддержать Блюдо "лайком" -*  пользователь имеет право поставить «лайк» блюду.
* *Добавить комментарий к блюду -* пользователь имеет право добавлять комментарии к блюдам.
* *Добавить Блюдо в "Избранное" -* пользователь имеет право добавлять блюда в список избранных.
* *Фильтр случайного Блюда из "Избранного" -* пользователь имеет право случайно подобрать блюдо из списка избранного.
* *Редактировать профиль -* пользователь имеет право изменять данные своего профиля.

**Действия неавторизированного пользователя в системе:**

* *Регистрация* – пользователь имеет право создать свой профиль.
* *Аутентификация* – пользователь имеет право войти в систему, если заранее создал профиль.
* *Просмотр комментариев к блюдам* - пользователь имеет право просматривать комментарии к блюдам.

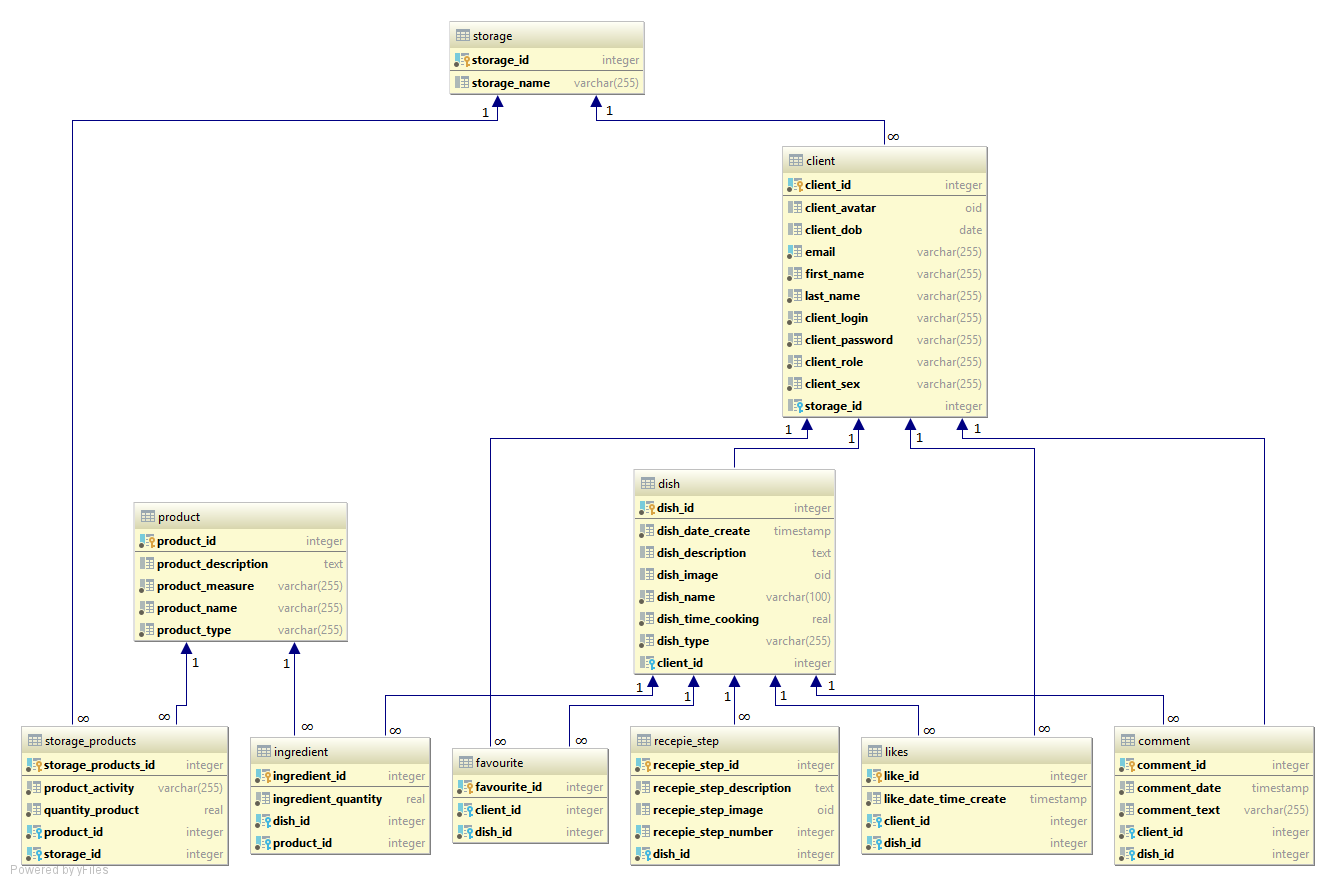
**Действия администратора в системе:**

* *Добавить/Изменить/Удалить все блюда* – пользователь имеет право создавать новые блюда, изменять и удалять все существующие блюда.
* *Добавить/Изменить/Удалить все продукты* – пользователь имеет право на создание, изменение и удаление все продуктов в системе.
* *Просмотр/Удаление Пользователей* – пользователь имеет право на просмотр и удаление всех пользователей в системе.
* *Просмотр/Добавление/Удаление комментариев к блюдам* – пользователь имеет право на добавление своих комментариев к блюдам, просмотр и удаление всех блюд в системе.

**Действия, разрешённые всем пользователям в системе:**

* *Просмотр всех блюд в системе –* все пользователи могут просмотреть все блюда.

**3.4.3. ER диаграмма базы данных веб приложения**



*Рисунок 3.5. ER диаграмма базы данных веб-приложения.*

**Таблица product - продукты:**

* *product\_id* – код продукта, первичный ключ
* *product\_description* – описание продукта.
* *product\_measure* – единица измерения продукта.
* *product\_name* – название продукта.
* *product\_type* – тип продукта.

**Таблица storage - холодильники:**

* *storage\_id* - код холодильника, первичный ключ.
* *storage\_name* – название холодильника.

**Таблица storage\_product – таблица связывающая продукты и холодильники:**

* *storage\_product\_id* – код, первичный ключи.
* *product\_activity* – активность продукта.
* *quantity\_product* – количество продукта.
* *product\_id* – код продукта, внешний ключ.
* *storage\_id* – код холодильника, внешний ключ.

**Таблица client – пользователь:**

* *client\_id* – код пользователя, первичный ключ.
* *client\_avatar* – изображение аккаунта пользователя.
* *client\_dob* – дата рождения пользователя.
* *email* – адрес электронной почты пользователя.
* *first\_name* – имя пользователя.
* *last\_name* – фамилия пользователя.
* *client\_login* – логин пользователя.
* *client\_password* – пароль пользователя.
* *client\_role* – роль пользователя.
* *client\_sex* – пол пользователя.
* *storage\_id* – код холодильника, внешний ключ.

**Таблица dish – блюдо:**

* *dish\_id* – код блюда, первичный ключ.
* *dish\_date*\_create – дата и время создания блюда.
* *dish\_description* – описание блюда.
* *dish\_image* – изображения блюда.
* *dish\_name* – названия блюда.
* *dish\_time\_cooking* – время приготовления блюда.
* *dish\_type* – тип блюда.
* *client\_id* – код автора блюда (пользователя).

**Таблица recipe\_step – шаг рецепта блюда**

* *recipe\_step\_id* – код шага рецепта, первичный ключ.
* *recipe\_step\_description* – описания шага рецепта.
* *recipe\_step\_image* – изображение шага рецепта.
* *recipe\_step\_number* – порядковый номер шага рецепта блюда.
* *dish\_id* – код блюда, внешний ключ.

**Таблица ingredient – ингредиент (таблица связывающая продукты и блюда):**

* *ingredient\_id* – код ингредиента, первичный ключ.
* *ingredient\_quantity* – количество продукта в блюде.
* *dish\_id* – код блюда, внешний ключ.
* *product\_id* – код продукта, внешний ключ.

**Таблица favorite – избранное (таблица связывающая пользователя и блюдо)**

* *favorite\_id* – код избранного, первичный ключ.
* *client\_id* – код пользователя, внешний ключ.
* *dish\_id* – код блюда, внешний ключ.

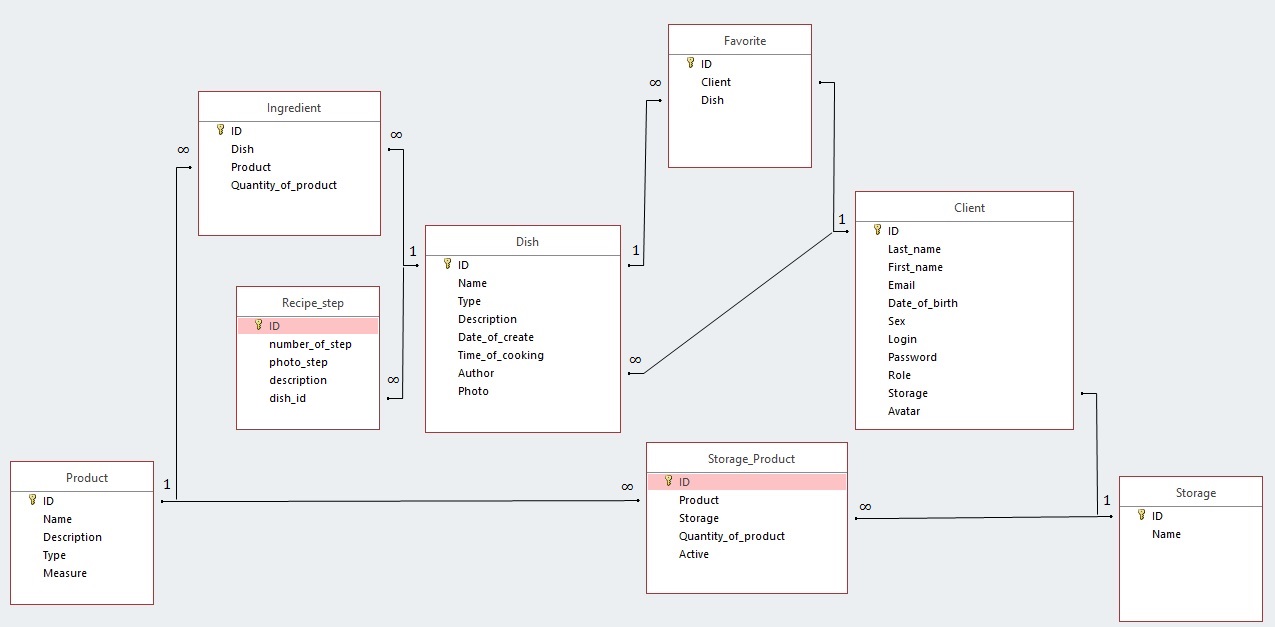
**Таблица like – понравившееся (таблица связывающая пользователя и блюдо)**

* *like\_id* – код понравившегося, первичный ключ.
* *like\_date\_time\_create* – дата и время создания понравившегося.
* *client\_id* – код пользователя, который создал понравившееся, внешний ключ.
* *dish\_id* – код блюда, которое понравилось, внешний ключ.

**Таблица comment – комментарий (таблица связывающая пользователя и блюдо)**

* *comment \_id* – код комментария.
* *comment \_date* – дата создания комментария.
* *comment\_text* – текс комментария.
* *client\_id* – код пользователя, который создал комментарий, внешний ключ.
* *dish\_id* – код блюда, которое прокомментировали , внешний ключ.

**3.4.4. ER диаграмма базы данных мобильного приложеня**



*Рисунок 3.6. ER диаграмма базы данным мобильного части приложения.*

**Таблица product - продукты:**

* *product\_id* – код продукта, первичный ключ.
* *product\_description* – описание продукта.
* *product\_measure* – единица измерения продукта.
* *product\_name* – название продукта.
* *product\_type* – тип продукта.

**Таблица storage - холодильники:**

* *storage\_id* - код холодильника, первичный ключ.
* *storage\_name* – название холодильника.

**Таблица storage\_product – таблица связывающая продукты и холодильники:**

* *storage\_product\_id* – код, первичный ключи.
* *product\_activity* – активность продукта.
* *quantity\_product* – количество продукта.
* *product\_id* – код продукта, внешний ключ.
* *storage\_id* – код холодильника, внешний ключ.

**Таблица client – пользователь:**

* *client\_id* – код пользователя, первичный ключ.
* *client\_avatar* – изображение аккаунта пользователя.
* *client\_dob* – дата рождения пользователя.
* *email* – адрес электронной почты пользователя.
* *first\_name* – имя пользователя.
* *last\_name* – фамилия пользователя.
* *client\_login* – логин пользователя.
* *client\_password* – пароль пользователя.
* *client\_role* – роль пользователя.
* *client\_sex* – пол пользователя.
* *storage\_id* – код холодильника, внешний ключ.

**Таблица dish – блюдо (Локальная БД будет содержать только блюда из таблицы Favorite):**

* *dish\_id* – код блюда, первичный ключ.
* *dish\_date*\_create – дата и время создания блюда.
* *dish\_description* – описание блюда.
* *dish\_image* – изображения блюда.
* *dish\_name* – названия блюда.
* *dish\_time\_cooking* – время приготовления блюда.
* *dish\_type* – тип блюда.
* *client\_id* – код автора блюда (пользователя).

**Таблица recipe\_step – шаг рецепта блюда (Локальная БД будет содержать шаги рецептов блюд только из таблицы Favorite)**

* *recipe\_step\_id* – код шага рецепта, первичный ключ.
* *recipe\_step\_description* – описания шага рецепта.
* *recipe\_step\_image* – изображение шага рецепта.
* *recipe\_step\_number* – порядковый номер шага рецепта блюда.
* *dish\_id* – код блюда, внешний ключ.

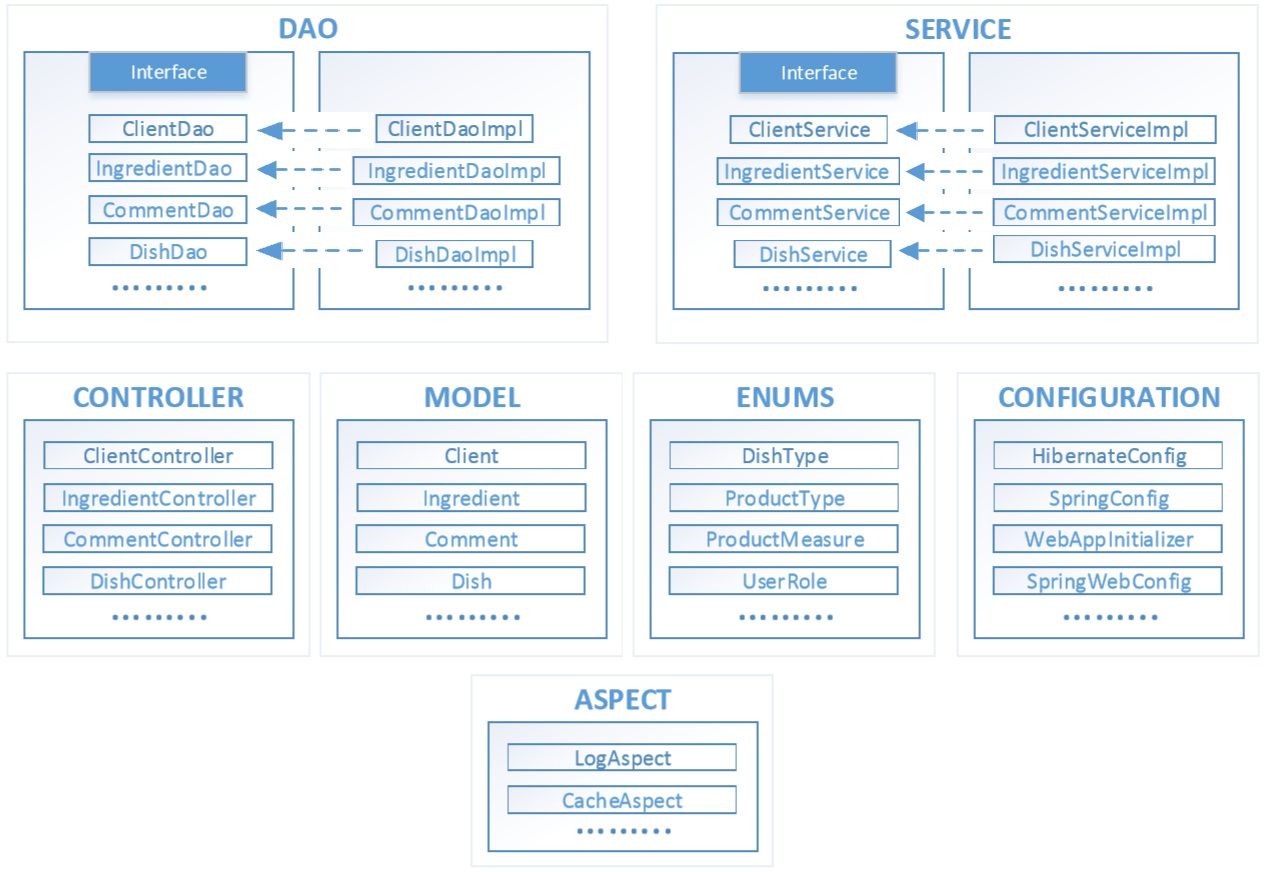
**Таблица ingredient – ингредиент (таблица связывающая продукты и блюда, локальная БД будет содержать ингредиенты только блюд из таблицы Favorite ):**

* *ingredient\_id* – код ингредиента, первичный ключ.
* *ingredient\_quantity* – количество продукта в блюде.
* *dish\_id* – код блюда, внешний ключ.
* *product\_id* – код продукта, внешний ключ.

**Таблица favorite – избранное (таблица связывающая пользователя и блюдо)**

* *favorite\_id* – код избранного, первичный ключ.
* *client\_id* – код пользователя, внешний ключ.
* *dish\_id* – код блюда, внешний ключ.

**3.4.5. Диаграмма классов**



*Рисунок 3.7. Диаграмма классов веб приложения.*

**MODEL**

Классы сущности, которые сохраняются в базу данных.

Client model

* *client\_id* – код пользователя, идентификационный номер.
* *client\_avatar* – изображение аккаунта пользователя.
* *client\_dob* – дата рождения пользователя.
* *email* – адрес электронной почты пользователя.
* *first\_name* – имя пользователя.
* *last\_name* – фамилия пользователя.
* *client\_login* – логин пользователя.
* *client\_password* – пароль пользователя.
* *client\_role* – роль пользователя.
* *client\_sex* – пол пользователя.
* *storage* – ссылка на модель Storage.

Ingredient Model

* *ingredient\_id* – код ингредиента, идентификационный номер.
* *ingredient\_quantity* – количество продукта в блюде.
* *dish* – ссылка на модель Dish.
* *product* –ссылка на модель Product.

**DAO(Data access object)**

Интерфейсы и их реализация, которые запрашивают и изменяют записи в базе данных.

Client Dao

* getAllClients() – запрашивает всех пользователей из базы данных.
* getClientById – запрашивает клиента по его идентификационному коду.
* addClient – добавляет нового пользователя в базу данных.
* updateClient – обновляет данные о пользователе в базу данных.
* deleteClient – удаляет пользователя из базы данных.
* getClientsByStorage – запрашивает список пользователей по холодильнику.
* getClientsByFavouriteDish – запрашивает список пользователей по блюду находящемуся в избранном .
* getClientByAuthoredDish – запрашивает автора блюда.
* getClientByEmail – запрашивает пользователя по email.
* getClientsByLikedDish – запрашивает список пользователей, которые поставили «лайк» блюду.
* getClientByComment – запрашивает пользователя по комментарию.
* getClientsByCommentedDish – запрашивает список пользователей, которые прокомментировали блюдо.

Пример реализации. Метод getClientsByCommentedDish, который запрашивает пользователя, которые прокомментировали блюдо.

**public** List<Client> getClientsByCommentedDish(Dish dish) {  
 **return** session()  
 .createQuery(**"select c.client from Comment as c where c.dishCommented = ?1"**, Client.**class**)  
 .setParameter(1, dish)  
 .list();  
}

Ingredient Dao

* getAllIngredients – запрашивает список всех ингредиентов.
* getIngredientById – запрашивает ингредиент по идентификационному номеру.
* getIngredientByDishAndProduct – запрашивает ингредиент по блюду и продукту.
* addIngredient – добавляет новый ингредиент в базу данных.
* updateIngredient – обновляет ингредиент в базе данных.
* deleteIngredient – удаляет ингредиент из базы данных .

**SERVICE**

Интерфейсы и их реализация, которая реализуют бизнес-логику приложения. Кэш данный происходит в этом лэйере.

Client Service

* getAllClients() – запрашивает всех пользователей из ClientDao.
* getClientById – запрашивает клиента по его идентификационному коду.
* addClient – добавляет нового пользователя .
* updateClient – обновляет данные о пользователе.
* deleteClient – удаляет пользователя.
* getClientsByStorage – запрашивает список пользователей по холодильнику.
* getClientsByFavouriteDish – запрашивает список пользователей по блюду находящемуся в избранном.
* getClientByAuthoredDish – запрашивает автора блюда.

Пример реализации. Метод getClientByAuthoredDish запрашивает автора блюда.

@Override  
**public** Client getClientByAuthoredDish(Dish authoredDish) {  
 **return clientDao**.getClientByAuthoredDish(authoredDish);  
}

* getClientByEmail – запрашивает пользователя по email.
* getClientsByLikedDish – запрашивает список пользователей, которые поставили «лайк» блюду.
* getClientByComment – запрашивает пользователя по комментарию.
* getClientsByCommentedDish – запрашивает список пользователей, которые прокомментировали блюдо.

Ingredient Dao

* getAllIngredients – запрашивает список всех ингредиентов.
* getIngredientById – запрашивает ингредиент по идентификационному номеру.
* getIngredientByDishAndProduct – запрашивает ингредиент по блюду и продукту.
* addIngredient – добавляет новый ингредиент.

**CONTROLLER**

Обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в JSON формате. Большая часть логов записываются в controller лэйере.

Client Controller

* getAllClients() – отправляет всех пользователей.
* getClientById – отправляет пользователя по идентификационному номеру.
* addClient – добавляет нового пользователя.
* updateClient – обновляет данные о пользователе.
* deleteClient – удаляет пользователя.
* getClientsByStorage – отправляет список пользователей по холодильнику.
* getClientsByFavouriteDish – отправляет список пользователей по блюду, находящемуся в избранном.
* getClientByAuthoredDish – отправляет автора блюда.
* getClientByEmail – отправляет пользователя по email.
* getClientsByLikedDish – отправляет список пользователей, которые поставили «лайк» блюду.
* getClientByComment – отправляет пользователя по комментарию.
* getClientsByCommentedDish – отправляет список пользователей, которые прокомментировали блюдо.

Пример реализации. Метод getClientsByCommentedDish отправляет список пользователей, которые прокомментировали блюдо.

@GetMapping(**"/clients/commented-dish/{id}"**)  
**public** ResponseEntity<List<Client>> getClientsByCommentedDish(@PathVariable **int** id){  
 **final** Dish dishCommented = **dishService**.getDishById(id);  
 **final** List<Client> clients = **clientService**.getClientsByCommentedDish(dishCommented);  
 **return new** ResponseEntity<>(clients, HttpStatus.***OK***);  
}

Ingredient Controller

* getAllIngredients – отправляет список всех ингредиентов.
* getIngredientById – отправляет ингредиент по идентификационному номеру.
* getIngredientByDishAndProduct – отправляет ингредиент по блюду и продукту.
* addIngredient – добавляет новый ингредиент.
* updateIngredient – обновляет ингредиент.
* deleteIngredient – удаляет ингредиент.

**ASPECT**

Реализация Spring AOP. Позволяет не расширять код связанный с бизнес логикой приложения в Service лэйер.

Log Aspect – аспект, который логирует все действия в системе

* logExceptions – логирует исключения выброшенные во время работы приложения.
* LogExecutionTime – логирует действия пользователей в системе.

Cache Aspect – аспект, который кэширует данные

* doCache – кэширует данные, для быстрого доступа к ним.

**ENUMS**

Вспомогательные классы для основных сущностей.

Dish Type – тип блюда

* SALAD - салат
* SNACK - закуски
* FIRST\_COURSE – первое блюдо
* SECOND\_COURSE – второе блюдо
* DOUGH - тесто
* SIDE\_DISH - гарнир
* SAUCE - соус
* DESERT - десерт
* BAKE - выпечка
* MEAT - мясные
* STRONG\_DRINKS – сильно алкогольные напитки
* SOFT\_DRINKS – слабоалкогольные напитки

Product Measure – мера продукта

* GRAM - граммы
* MILLIGRAM - миллиграммы
* KILOGRAM - килограмм
* LITER - литр
* MILLILITER- миллилитр
* UNIT - единица

**CONFIGURATIONS**

Классы, конфигурирующие Hibernate и Spring.

Hibernate Config

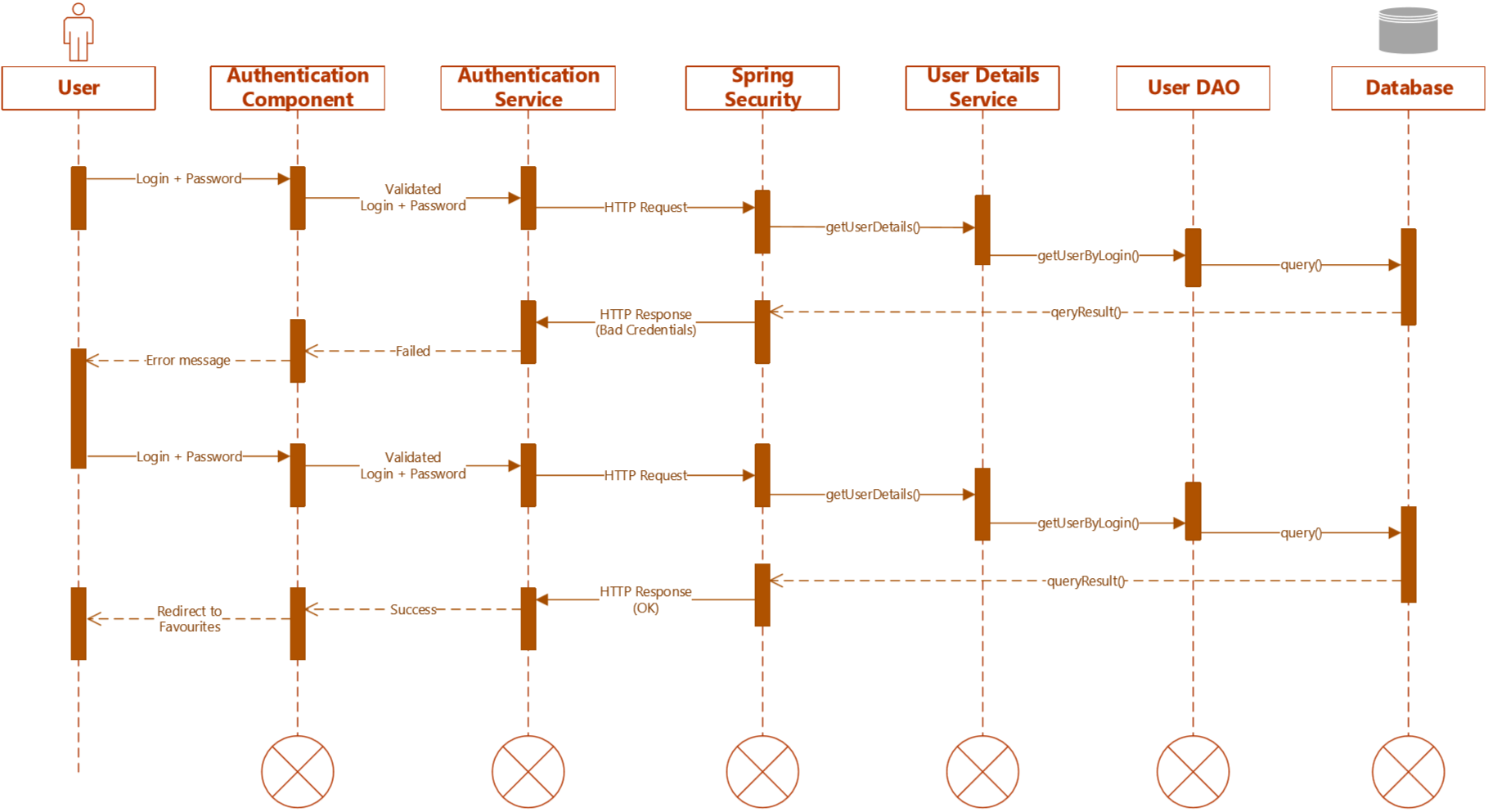
* dataSource – указывает тип базы данных, драйвер к нему, схема в базе данных , логин и пароль пользователя базы данных.
* sessionFactory – соединяет dataSource, какие пакеты просканировать для связки сущностей и hibernateProperties
* transactionManager – для транзакции указываем sessionFactory
* hibernateProperties – настройки hibernate, диалект и др.

Пример кода. Метод hibernateProperties, указываются настройки для Hibernate

**private** Properties hibernateProperties()  
{  
 **final** Properties properties = **new** Properties();  
 properties.put(**"hibernate.dialect"**,**"org.hibernate.dialect.PostgreSQLDialect"**);  
 properties.put(**"hibernate.show\_sql"**,**"true"**);  
 properties.put(**"hibernate.format\_sql"**,**"true"**);  
 properties.put(**"hibernate.hbm2ddl.auto"**, **"update"**);  
 **return** properties;  
}

**3.4.6. Диаграмма прецедентов использования (Sequence diagram)**

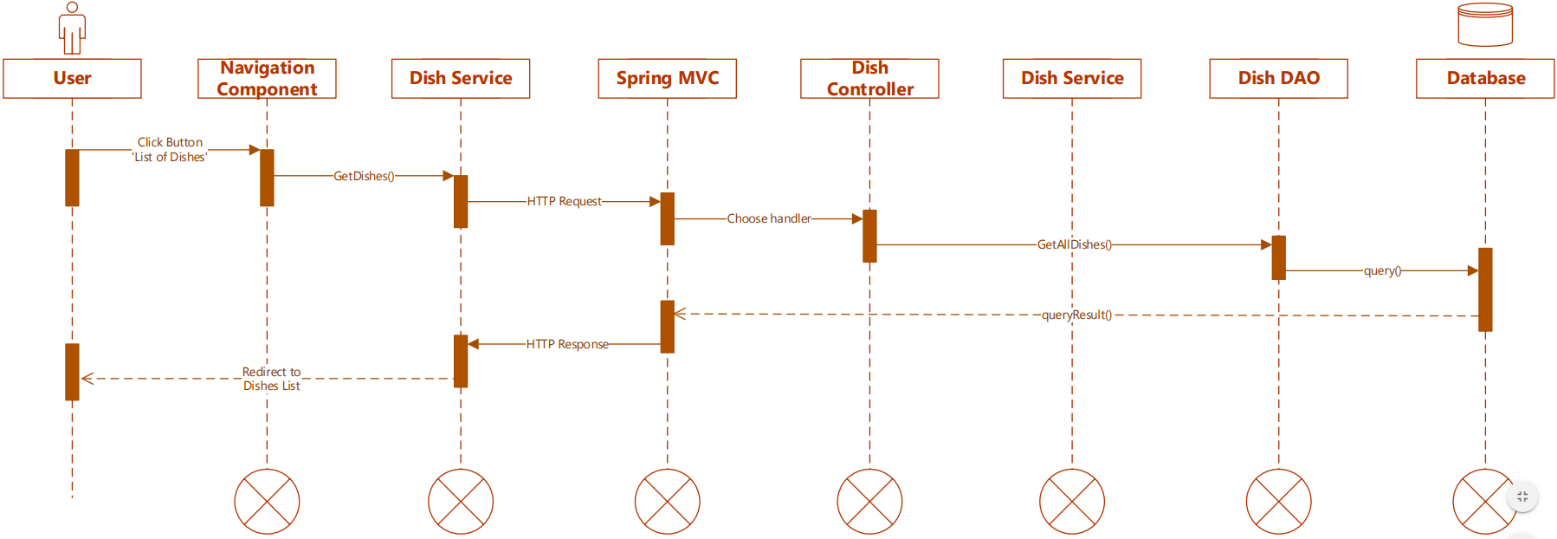
**Диаграмма прецедентов использования для сценария аутентификации**



*Рисунок 3.8. Диаграмма прецедентов использования для сценария аутентификации.*

Пользователь вводит неверный Логин или Пароль нажимает на кнопку «Войти». Authentication Service (сервис аутентификации) – проверяет логин и пароль на правильность введенных символов и затем отправляет на сервер через HTTP Request. Spring Security перехватывает данные и запрашивает у User Details Service пользователя по логину и сверяет с полученными данными из HTTP Request. В случае неверных данных возвращает HTTP 401 Bad Credentials, иначе возвращает HTTP 200 OK и перенаправляет на главную страницу.

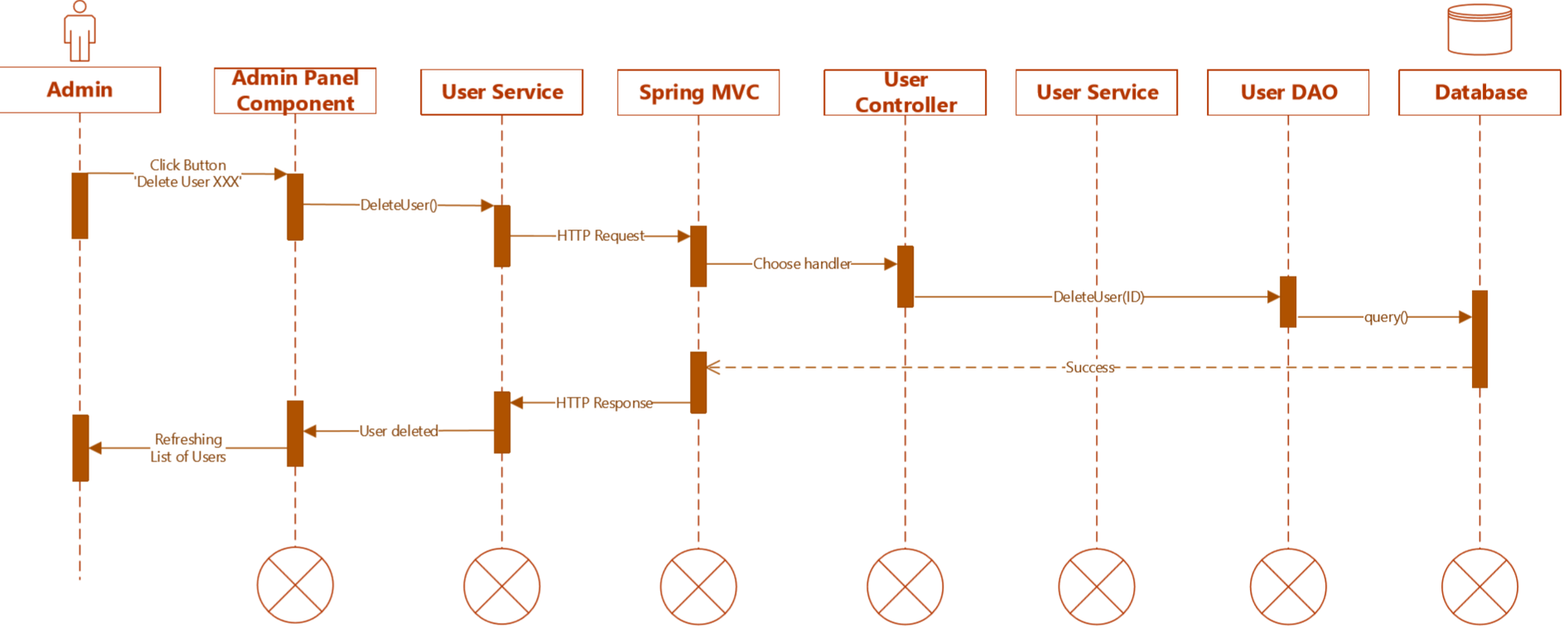
**Диаграмма прецедентов использования для сценария вывода всех блюд.**



*Рисунок 3.9. Диаграмма прецедентов использования для сценария всех блюд.*

Пользователь нажимает на кнопку «Список блюд». Dish Service(сервис блюд) отправляет HTTP Request. Spring MVC принимает запрос по URL и направляет к контроллеру отвечающему за блюда (Dish Controller). Контроллер запрашивает у Dish Service список блюд ( GetAllDishes() ) . Данный запрос проходит через DishDAO() - который предоставляет абстрактный интерфейс к какому-либо типу базы данных или механизму хранения и возвращает список блюд через HTTP Response. Пользователя перенаправляет на страницу со списком блюд.

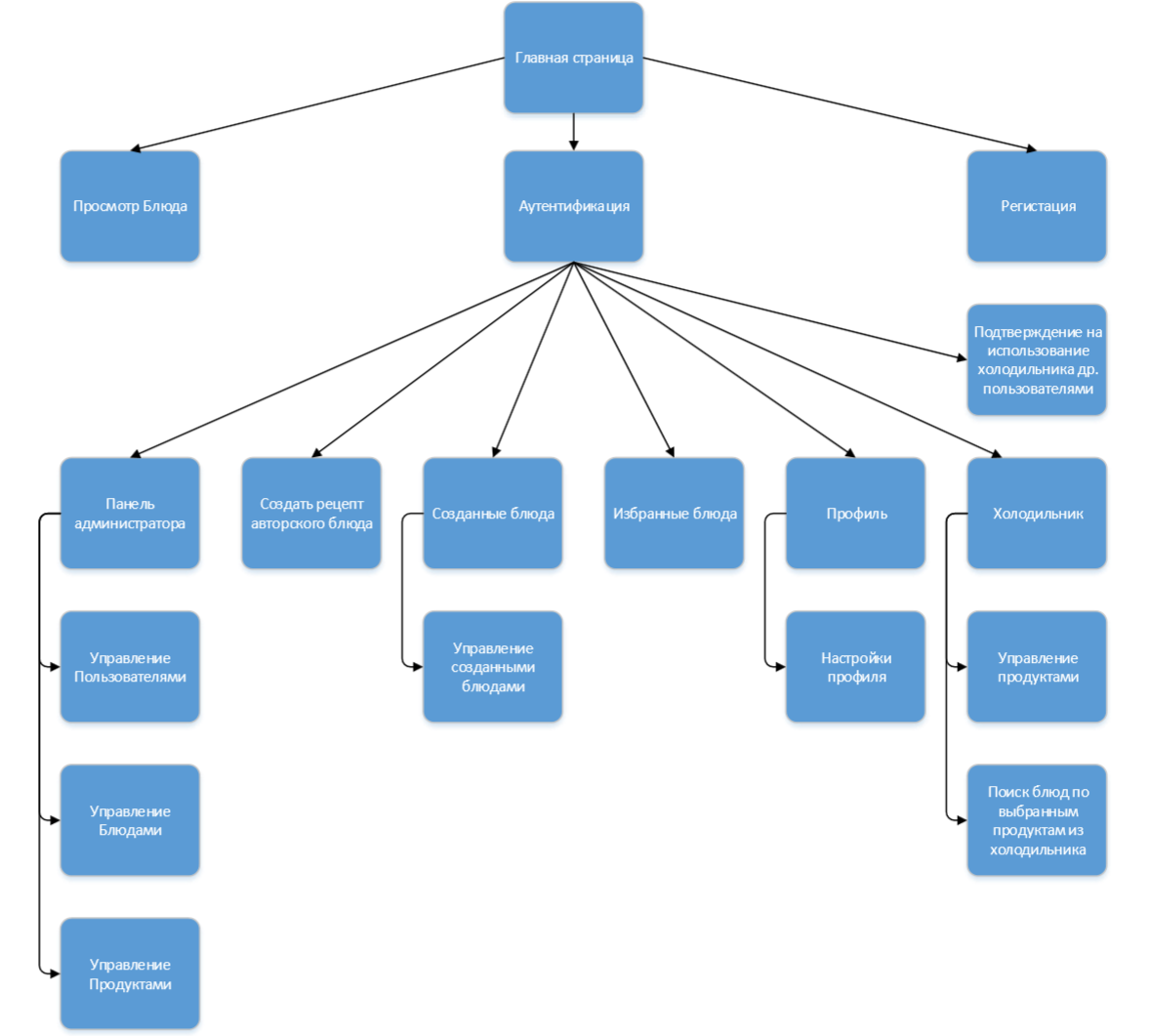
**Диаграмма прецедентов использования для сценария удаления пользователя.**



*Рисунок 3.10. Диаграмма прецедентов использования для сценария удаления пользователя.*

Пользователь Администратор (Admin) – нажимает на кнопку «Удалить Пользователя». Данное событие отправляет HTTP Request из User Service. Spring MVC принимает запрос по URL и направляет к контроллеру отвечающему за пользователей (User Controller). Контроллер запрашивает удаление пользователя по его ID у User Service ( DeleteUser(ID) ) . Данный запрос проходит через User DAO () - который предоставляет абстрактный интерфейс к какому-либо типу базы данных или механизму хранения и возвращает ответ о удачном удаление пользователя через HTTP Response и Администратору высвечивается данное сообщение.

**3.4.7. Дерево интерфейсов**



*Рисунок 3.11. Дерево интерфейсов веб приложения*

**Главная страница.**



Страница содержит список всех блюд с фильтром по категориям.

Переход на страницы :

Неавторизированного пользователя:

* + Просмотр блюда
  + Аутентификация
  + Регистрация

Авторизированного пользователя:

Обычный пользователь:

* + - Создать рецепт авторского блюда
    - Созданные блюда
    - Избранные блюда
    - Профиль
    - Холодильник
    - Подтверждение на использование холодильника другими пользователями
    - Просмотр блюда

Администратор:

* + - Панель администратора

**Регистрация**



Создать профиль в системе для использования более обширного функционала.

**Просмотр блюда**



Подробная информация о блюде. Ингредиенты и пошаговый рецепт приготовления. Авторизированные пользователи могу добавить блюдо в избранное, поставить «лайк» и прокомментировать. Администраторы могут удалять комментарии.

**Панель администратора**



Пользователи администраторы могут управлять Пользователями(создавать/удалять), Блюдами(создавать/ изменять неавторские блюда/ удалять все блюда) и Продуктами(создавать/ изменять/ удалять).

**Создать авторское блюдо**



Форма создания авторского блюда. Ввод данных о блюде его рецепте и ингредиентах с фотографиями.

**Созданные блюда**



Авторизированный пользователь может управлять созданными ранее авторскими блюдами (изменение/удаление)

**Избранные блюда**



Просмотр блюд, которые ранее были добавлены в избранное.

**Профиль**



Просмотр информации о пользовательском профиле. Изменение своего профиля.

**Холодильник**



Управление холодильником (добавление, изменение количества, удаление и деактивация продуктов). Поиск наиболее подходящих блюд из добавленных продуктов.