МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Система управления рестораном

Курсовая работа по дисциплине «Технологии программирования»

09.03.04 *Программная инженерия*

Обучающийся студент 3 курса М. В. Злобин

Обучающийся студент 3 курса А. Н. Пожидаев

Преподаватель В.С. Тарасов

Воронеж 2020

Оглавление

Введение

В современном мире тяжело представить свою жизнь без ресторанов. К сожалению, не все заведения работают корректно, и их владельцы не в состоянии исправить это самостоятельно, поэтому они вынуждены искать какую-либо систему для оптимизации. Однако, для самих ресторанов не существует системы, направленной на удобное распределение человеческих ресурсов, или они слишком дорогие.

Таким образом, целью нашей работы является создание такой системы, которая позволит управлять снабжением ресторана и оформлением заказов.

В свою очередь, работники ресторана смогут просматривать новые заказы и получать информацию о них.

Постановка задачи

Цель: автоматизировать работу персонала внутри организации.

Сфера применения: ресторан.

Требования к системе в целом:

- Система должна быть реализована по клиент-серверной архитектуре на технологиях ASP NET API и Angular любых версий
- Система должна использовать MSSQL
- Система не должна позволять доступ неавторизированным пользователям доступ к данным системы
- Система должна предоставлять физический доступ к интерфейсу только той роли, под которой вошел в систему пользователь

Задачи:

1. Провести анализ рынка с целью выявления достоинств и недостатков схожих по функционалу систем
2. Спроектировать приложение с учетом информации, полученной ранее в ходе анализа
3. Реализовать прототип приложения, обладающий функционалом, описанным в требованиях
4. Описать процесс разработки и результат

Описание функционала системы

Система управления рестораном представляет собой веб-приложение с ролевым доступом к компонентам. В системе должны быть предусмотрены следующие роли. В системе предусмотрены следующие роли:

- Администратор
- Повар
- Официант
- Провизор

Основные функциональные возможности Системы:

- Зарегистрировать сотрудника
- Настройка и просмотр меню
- Закупка ингредиентов и внесение их в систему
- Получение статистики о выручке, проданных блюдах
- Просмотр ингредиентов текущих и необходимых для закупки. Считать, что какой-либо продукт необходимо закупить если у оставшихся продуктов срок годности меньше недели или ингредиентов недостаточно для приготовления 10 порций блюда
- Составление заказа и пометка его готовым
- Оповещения о готовности заказов

1. Анализ предметной области

В современном мире мы наблюдаем проникновение информационных технологий практически во все сферы человеческой деятельности. Сейчас система автоматизации работы ресторана, бара и кафе - это не модное новшество, а необходимое условие конкурентоспособности всего бизнеса. Без специальных программ не получится организовать контроль над работой персонала, продажи не поднимутся, а издержки будут постоянно расти. Все процессы, которые происходят в заведении, нужно отслеживать. Важно соединить в одном софте деятельность кухни и зала, а также учесть возможность просматривать статистику.

Установка ПО сделает проще жизнь работникам и руководству.

1.1.Сравнение с аналогами

Среди аналогов, направленных на автоматизацию ресторанов/кафе/баров, можно выделить следующие приложения:

1. **Трактиръ**
   (https://traktir.ru/production/)

Программа, включает в себя целую линейку разнообразных продуктов. Автоматизирует процесс обслуживания, помогает управлять основной деятельностью, вести бухучет и руководить бизнесом.

Минусами этой программы можно считать:

- Сложный и непонятный интерфейс.
- Высокая стоимость.
- Отсутствие связи между поварами и официантами.

Плюсами этой программы можно считать:

- Отслеживает факты злоупотребления персонала.
- Интегрируется с 1С.

2. **Cafe Manager**
   (https://cafe-manager.ru/ **)**

Еще один облачный сервис, помогающий контролировать работу персонала, управлять им. Доступна настройка под потребности конкретного кафе. Возможность оперативно добавлять товары в статистику, планировать дальнейшее развитие. Войти в систему можно с любого планшета, ПК или телефона с помощью логина и пароля. Подходит для любого заведения.

Минусами этой программы можно считать:

- В случае отсутствия интернета могут появиться сбои.
- Высокая стоимость
- Нет хранилища файлов.

Плюсами этой программы можно считать:

- Доступен в 99% случаев.
- Можно заказать разработку функционала под требования конкретного заведения.

Так же есть много других сервисов, которые представляют клиентам разнообразные системы под их заведения.:

1. **R-Keeper**
   (http://restaurants.rkeeper.ru/)

2. **Iiko**
   (https://iiko.ru/)

3. **Tillypad**
   (https://tillypad.ru/)

4. **Quick Resto**
   (https://quickresto.ru/)

5. **Poster**
   (https://joinposter.com/)

6. **Ivideon**
   (https://ru.ivideon.com/)

Перечисленные сервисы предусматривают работу системы, как для владельцев, так и для самих работников, но являются платными решениями и не всегда помогут удовлетворить владельцев нужным им функционалом.

1.2 Анализ задач

Подробное описание всех прецедентов (пункт 4) и таблица ролевого доступа к компонентам (пункт 3.1) описаны в Настоящем Техническом Задании. Вход в систему, то есть авторизация, будет осуществляться с помощью JWT токена (JSON Wen Token), так как это самый популярный и эффективный способ авторизации для клиент-серверной архитектуры. Ее смысл состоит в том, чтобы на стороне сервера проверить введенный логин и пароль пользователя, далее сгенерировать уникальный токен доступа, подписанный секретным ключом, и передать его клиенту. Клиент же в свою очередь должен использовать возвращенный ему токен в заголовках в каждом своем запросе на сервер, чтобы сервер мог идентифицировать текущего пользователя. Графически это изображено на диаграмме активности для прецедента «Вход в систему».

1.3 Графическое описание работы системы

Диаграмма прецедентов

Более подробно диаграмму можно рассмотреть на доске в Miro.

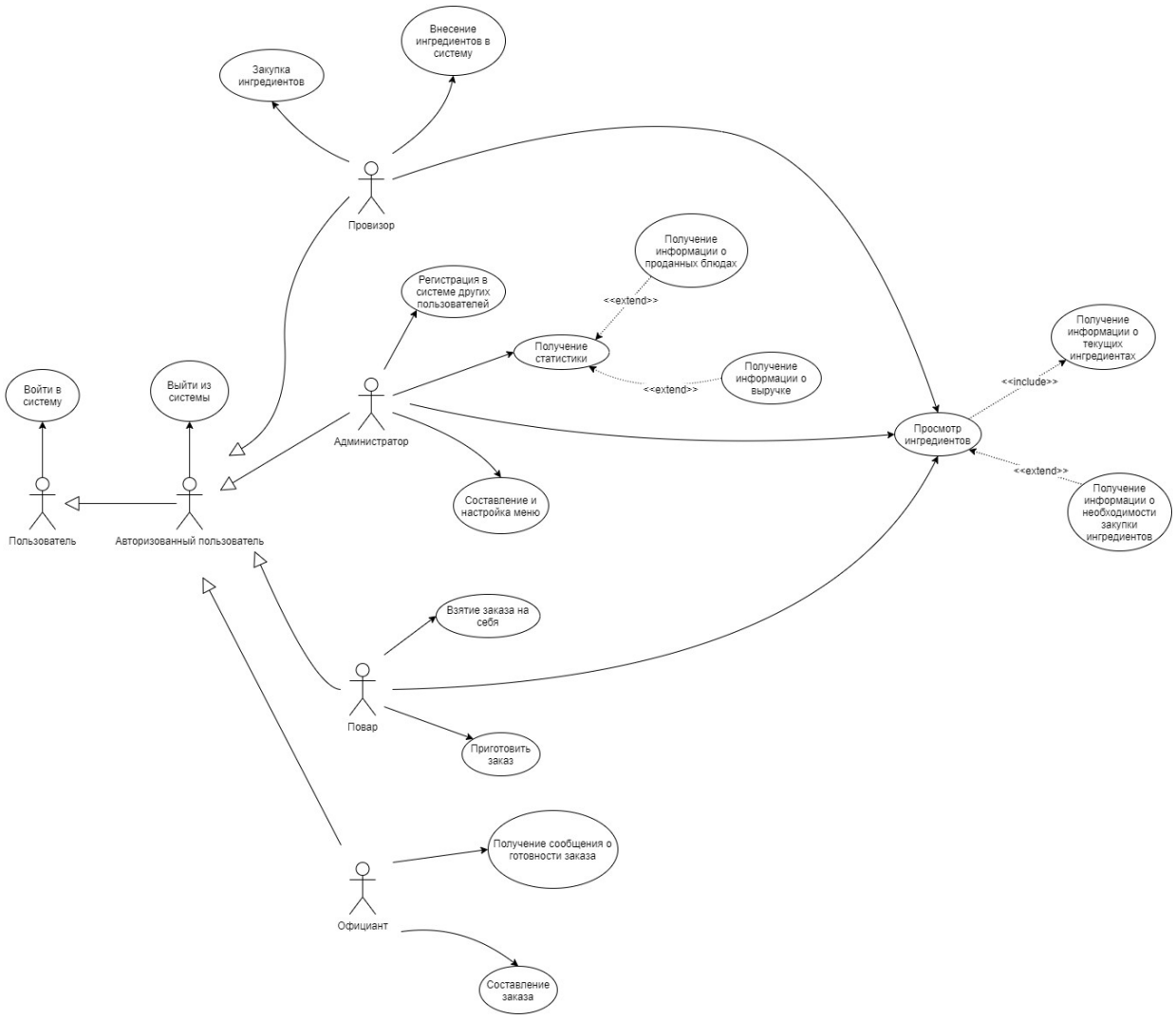


Диаграмма последовательностей

Диаграмма последовательностей для прецедента «Добавления ингредиентов». Подразумевается, что пользователь уже вошел в систему.

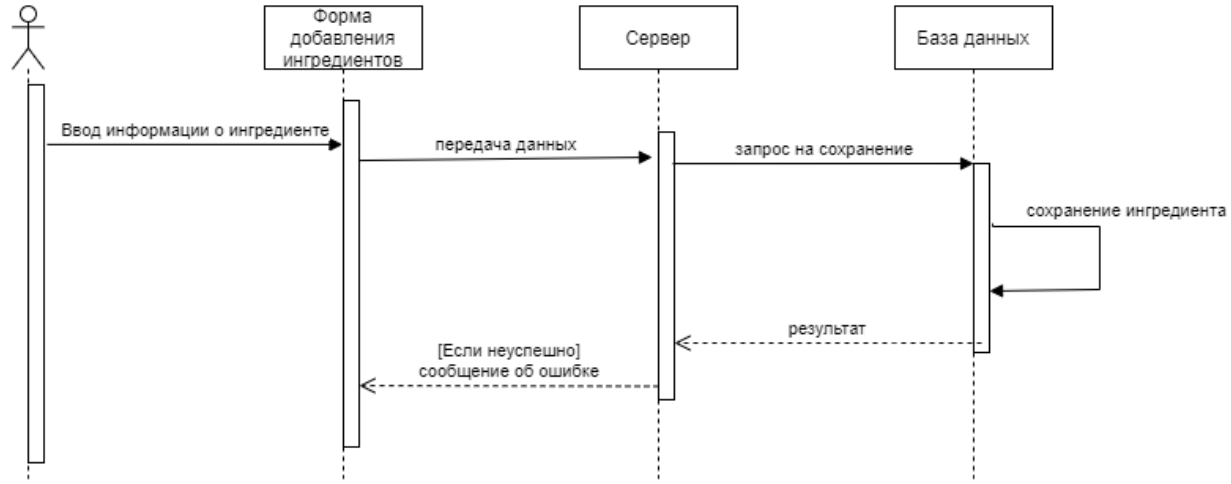


Рисунок 1 - Диаграмма последовательностей для Добавления ингредиентов

Диаграмма последовательностей для прецедента «Закупки ингредиентов». Подразумевается, что пользователь уже вошел в систему.

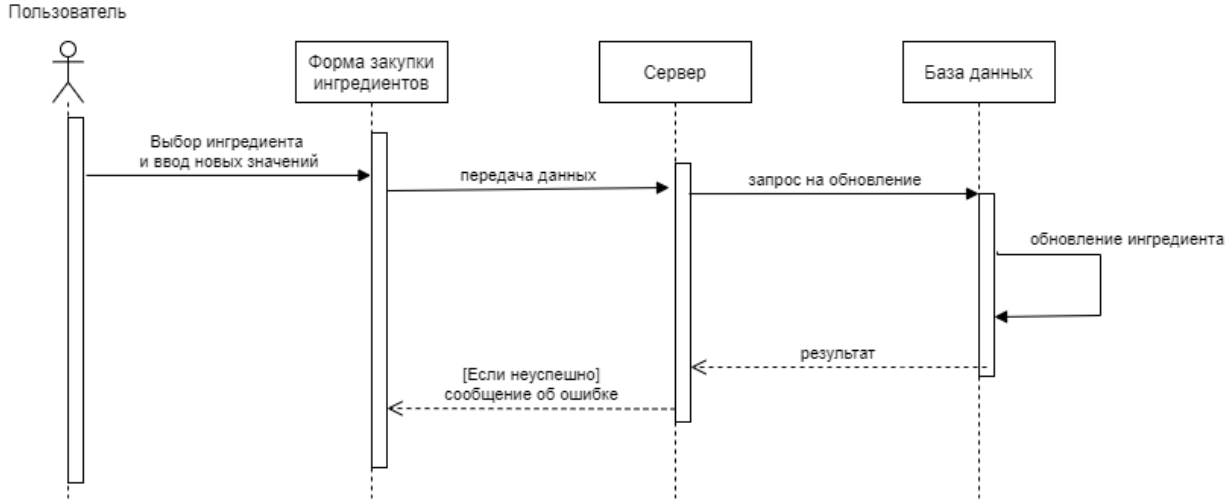


Рисунок 2 - Диаграмма последовательностей для Закупки ингредиентов

Диаграмма последовательностей для прецедента «Регистрация сотрудника». Подразумевается, что пользователь уже вошел в систему.

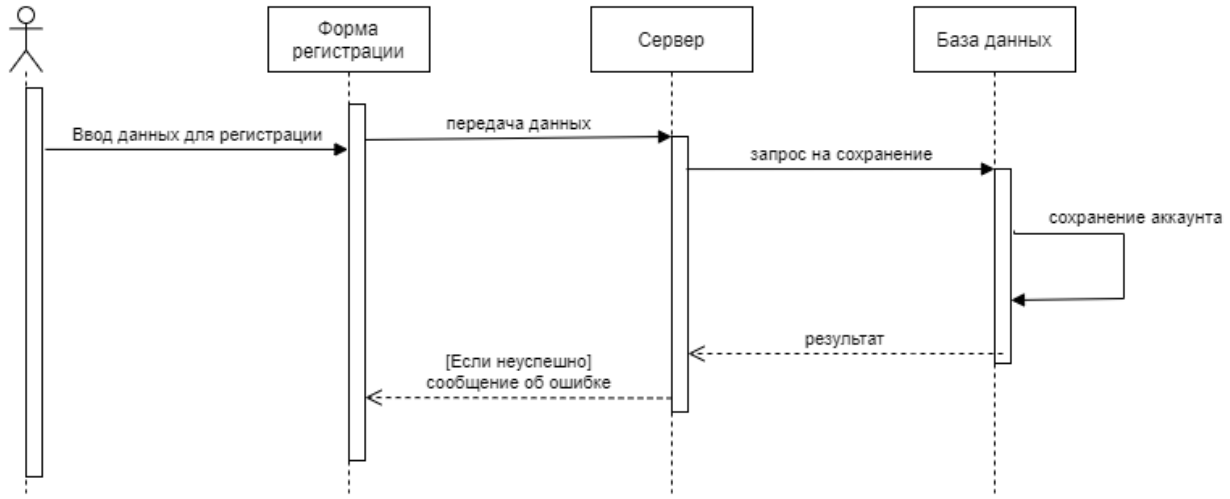


Рисунок 3 диаграмма последовательностей для Регистрации

Диаграмма последовательностей для прецедента «Создание заказа». Подразумевается, что пользователь уже вошел в систему.

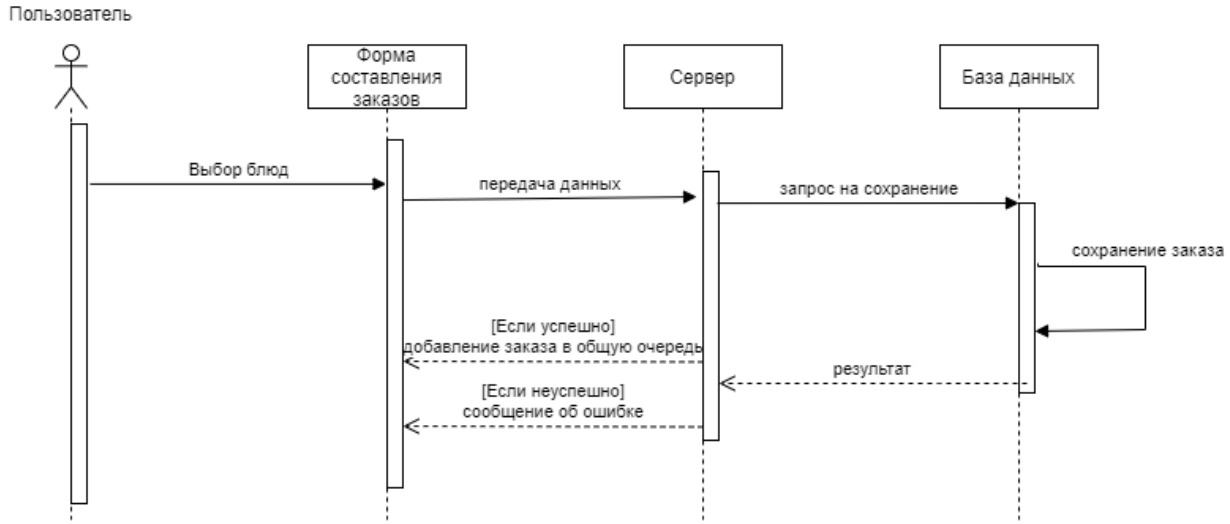


Рисунок 4 диаграмма последовательностей для Создания заказов

Диаграмма последовательностей для прецедента «Настройка меню». Подразумевается, что пользователь уже вошел в систему.

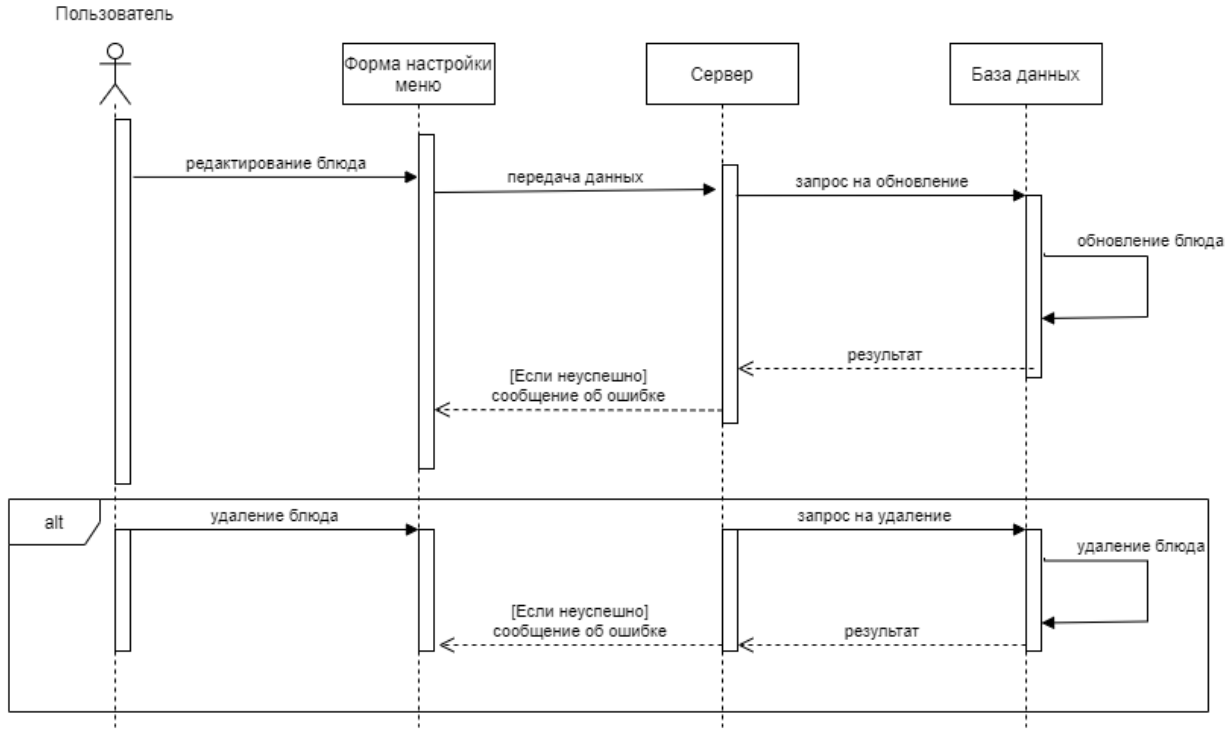


Рисунок 5 - диаграмма последовательностей для Настройки меню

Диаграмма последовательностей для прецедента «Взять заказ». Подразумевается, что пользователь уже вошел в систему.

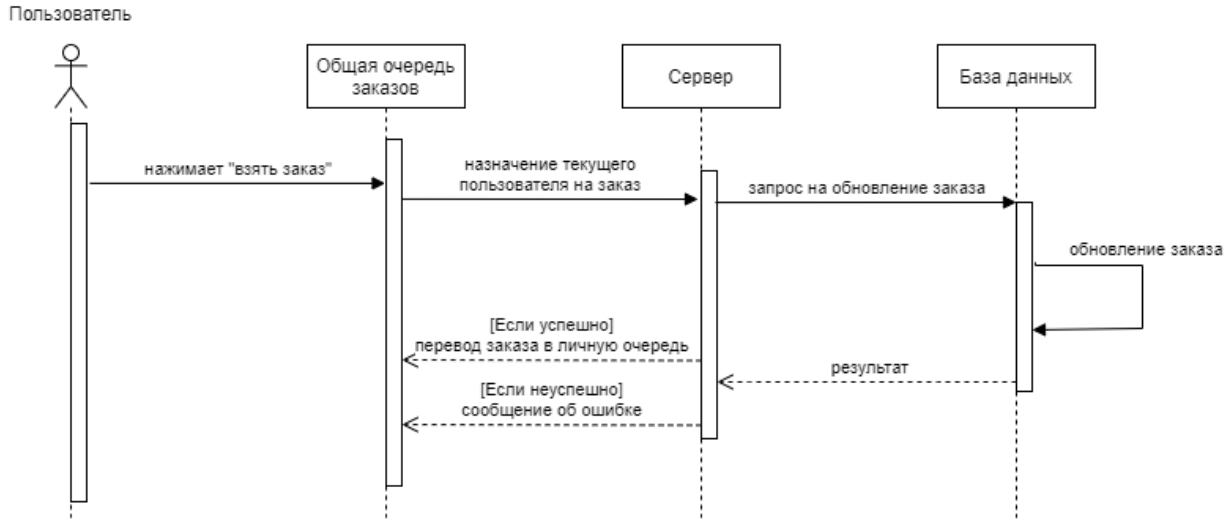


Диаграмма последовательностей для прецедента «Приготовить заказ». Подразумевается, что пользователь уже вошел в систему.

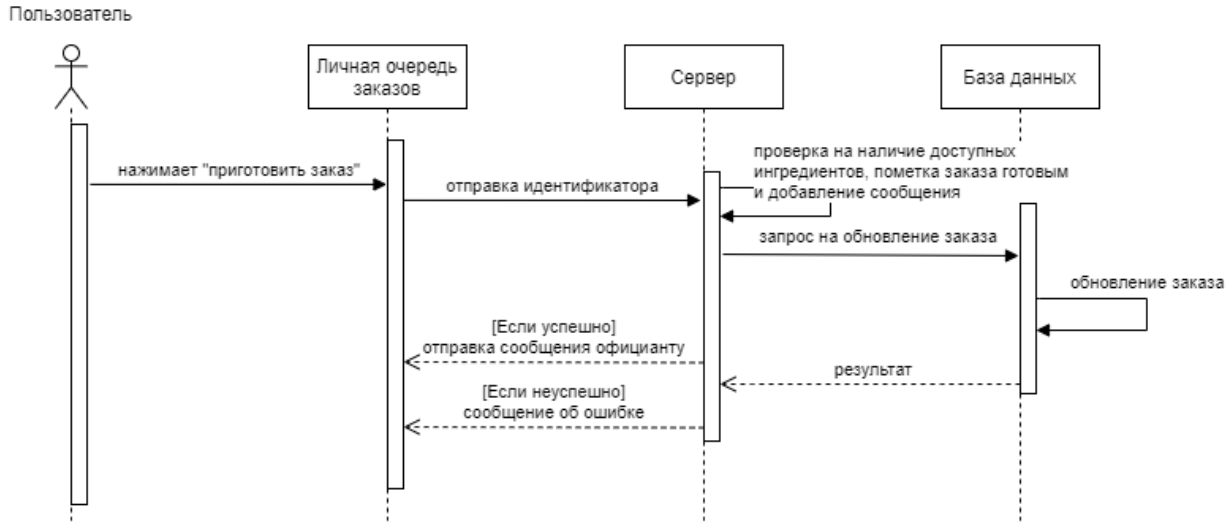


Диаграмма последовательностей для прецедентов «Получение статистики» и «Просмотр ингредиентов». Подразумевается, что пользователь уже вошел в систему.

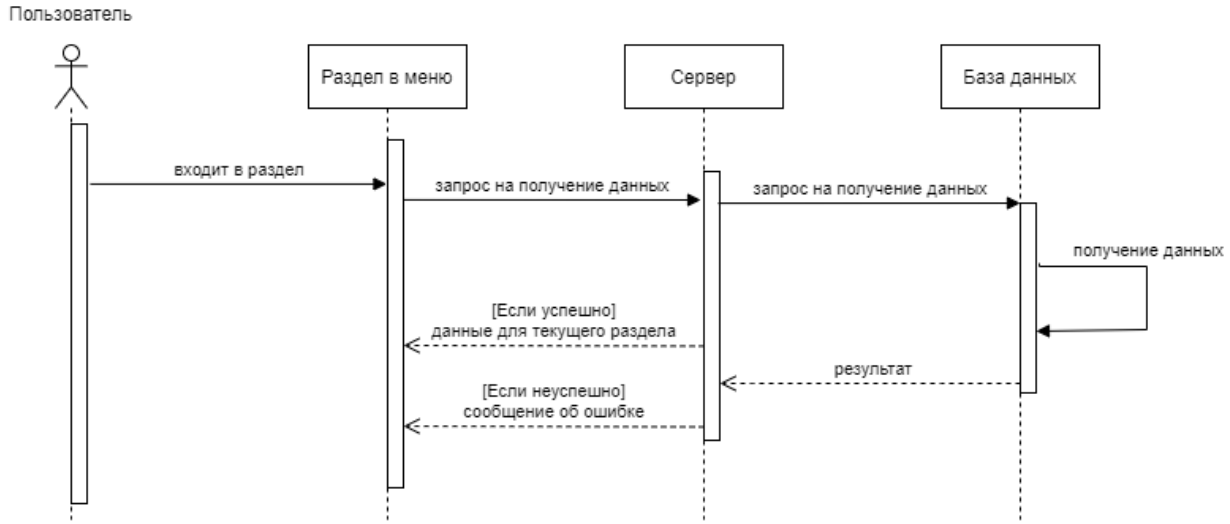


Диаграмма коммуникаций

Диаграмма коммуникаций для прецедента «Взять заказ». Подразумевается, что пользователь уже вошел в систему.

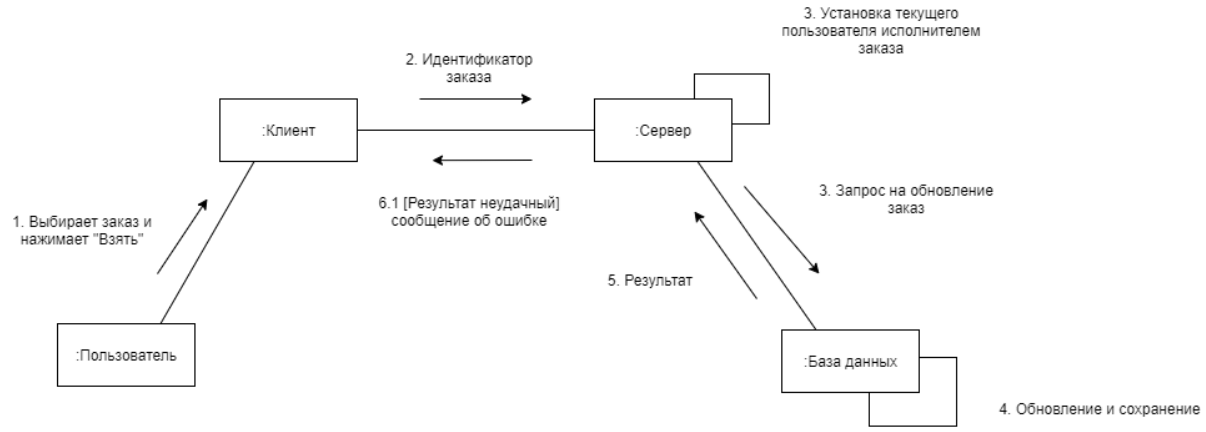


Диаграмма коммуникаций для прецедента «Приготовить заказ». Подразумевается, что пользователь уже вошел в систему.

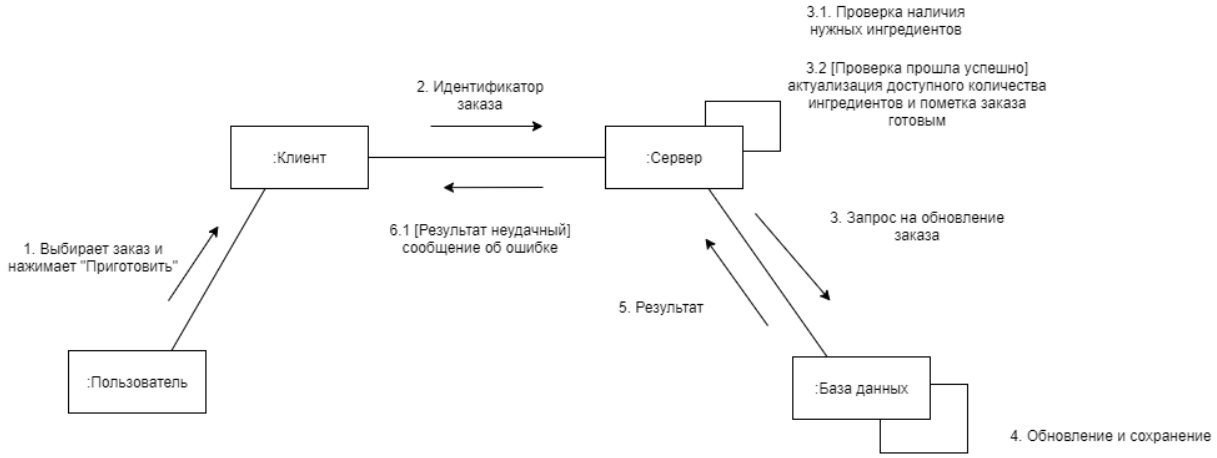


Диаграмма коммуникаций для прецедентов «Добавить ингредиент», «Создать заказ», «Регистрация сотрудника». Подразумевается, что пользователь уже вошел в систему.

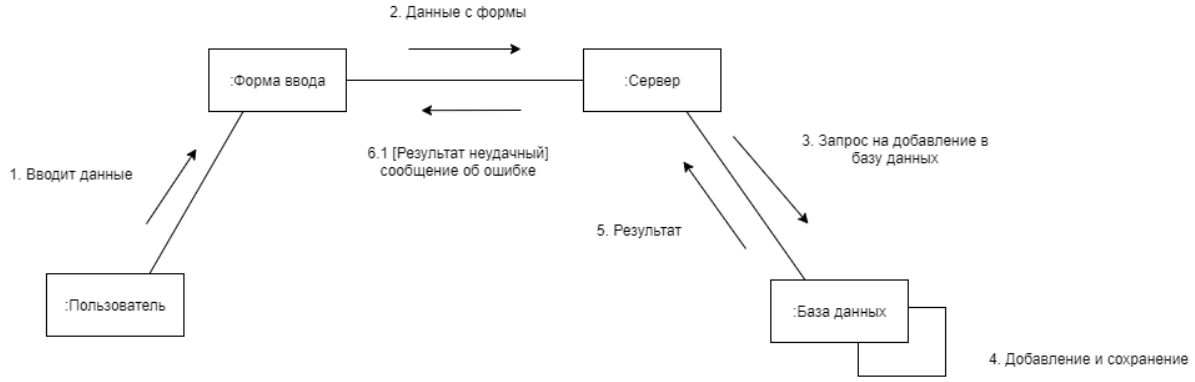


Диаграмма коммуникаций для прецедентов «Настройка меню», «Закупка ингредиентов». Подразумевается, что пользователь уже вошел в систему.

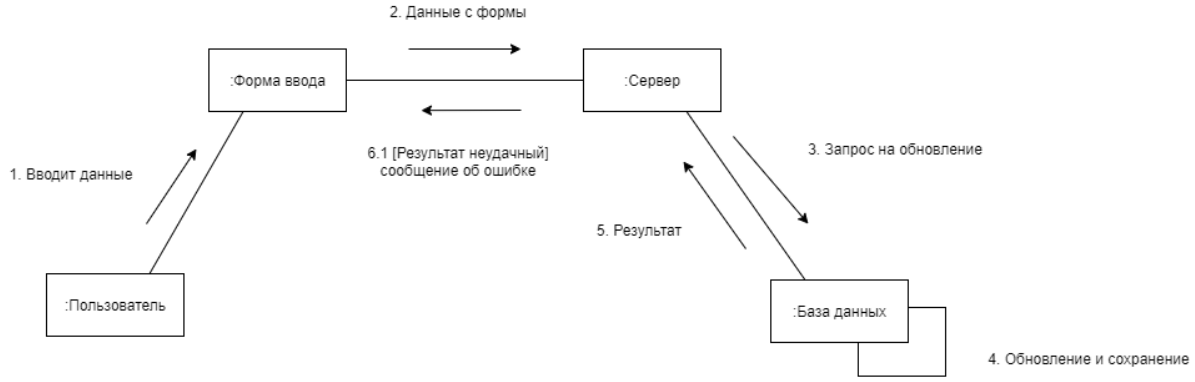


Диаграмма состояний

Более подробно диаграмму можно рассмотреть на доске в Miro.

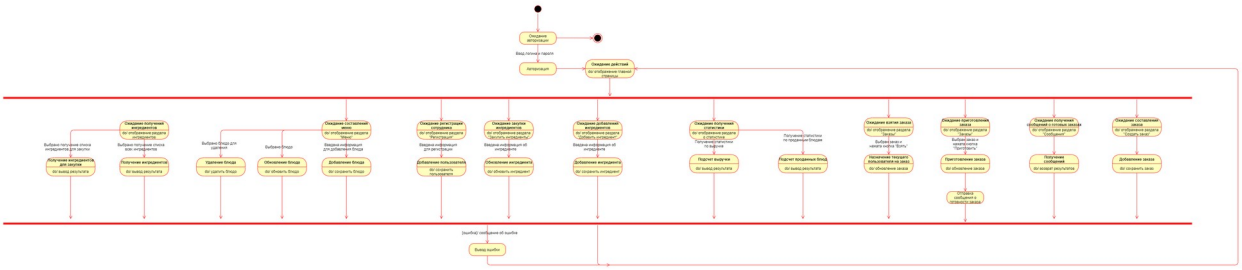


Диаграмма активности

Диаграмма активности для прецедента «Вход в систему» (Авторизация).

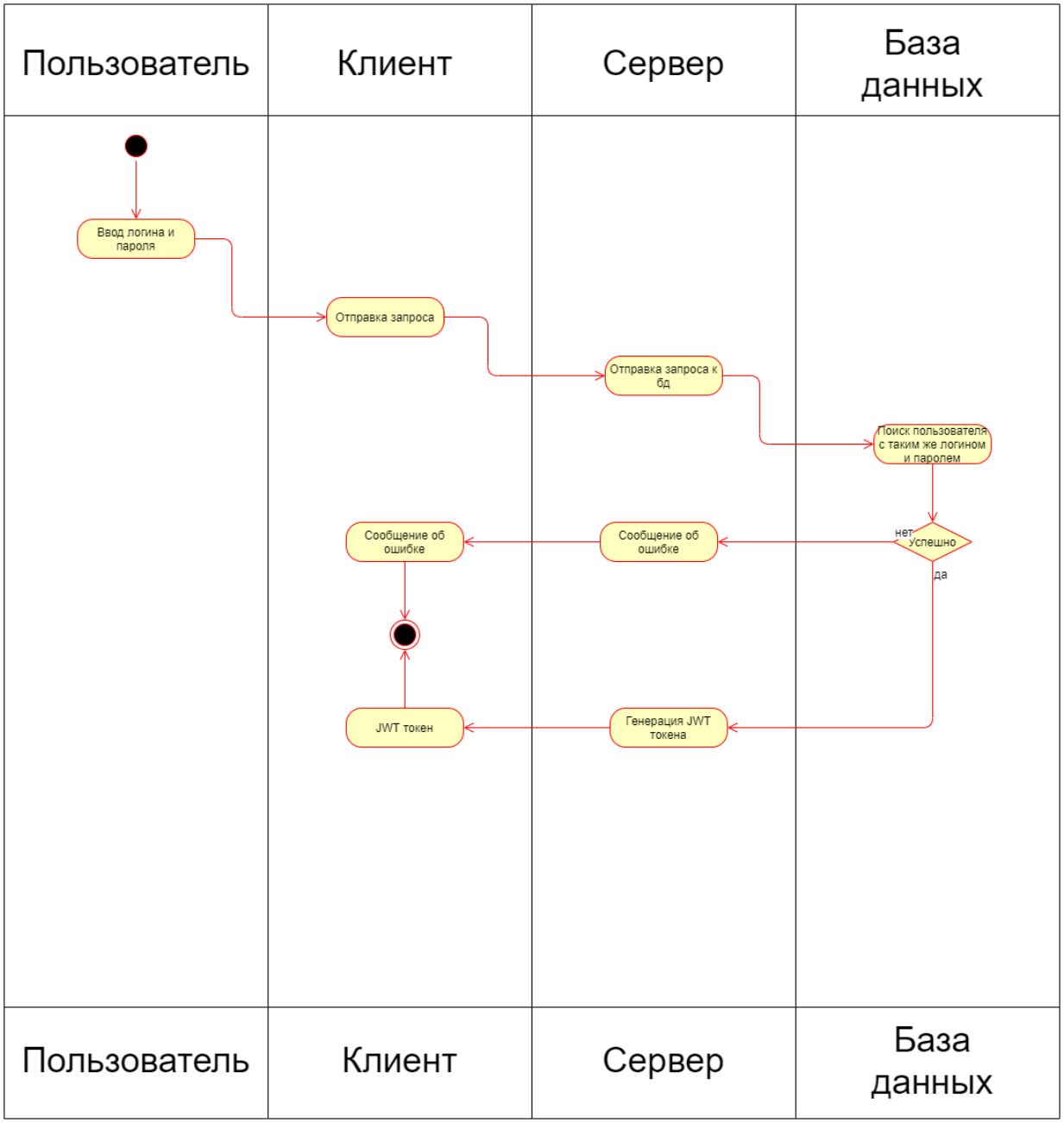


Диаграмма активности для прецедента «Взять заказ».

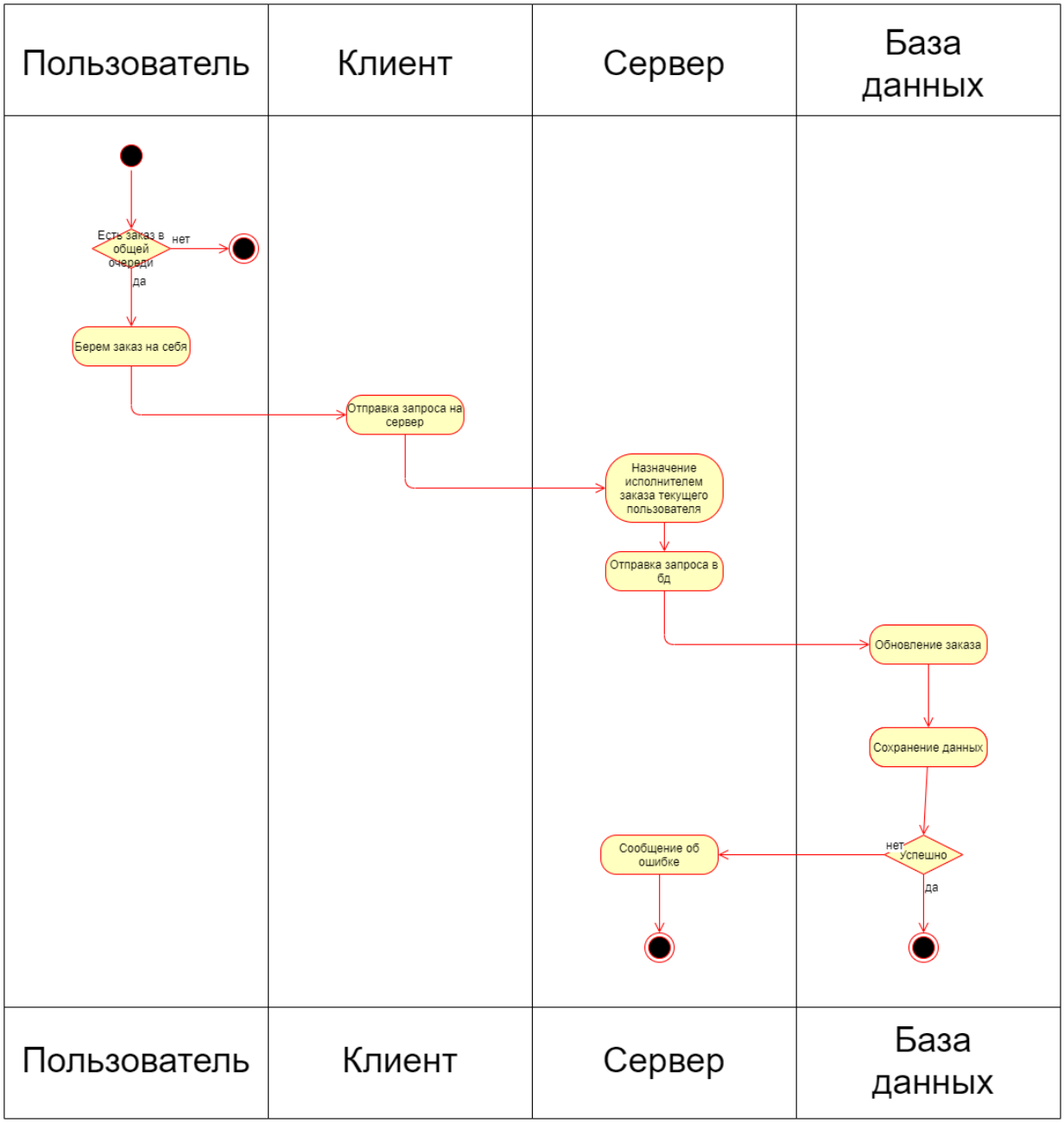


Диаграмма активности для прецедента «Приготовить заказ».

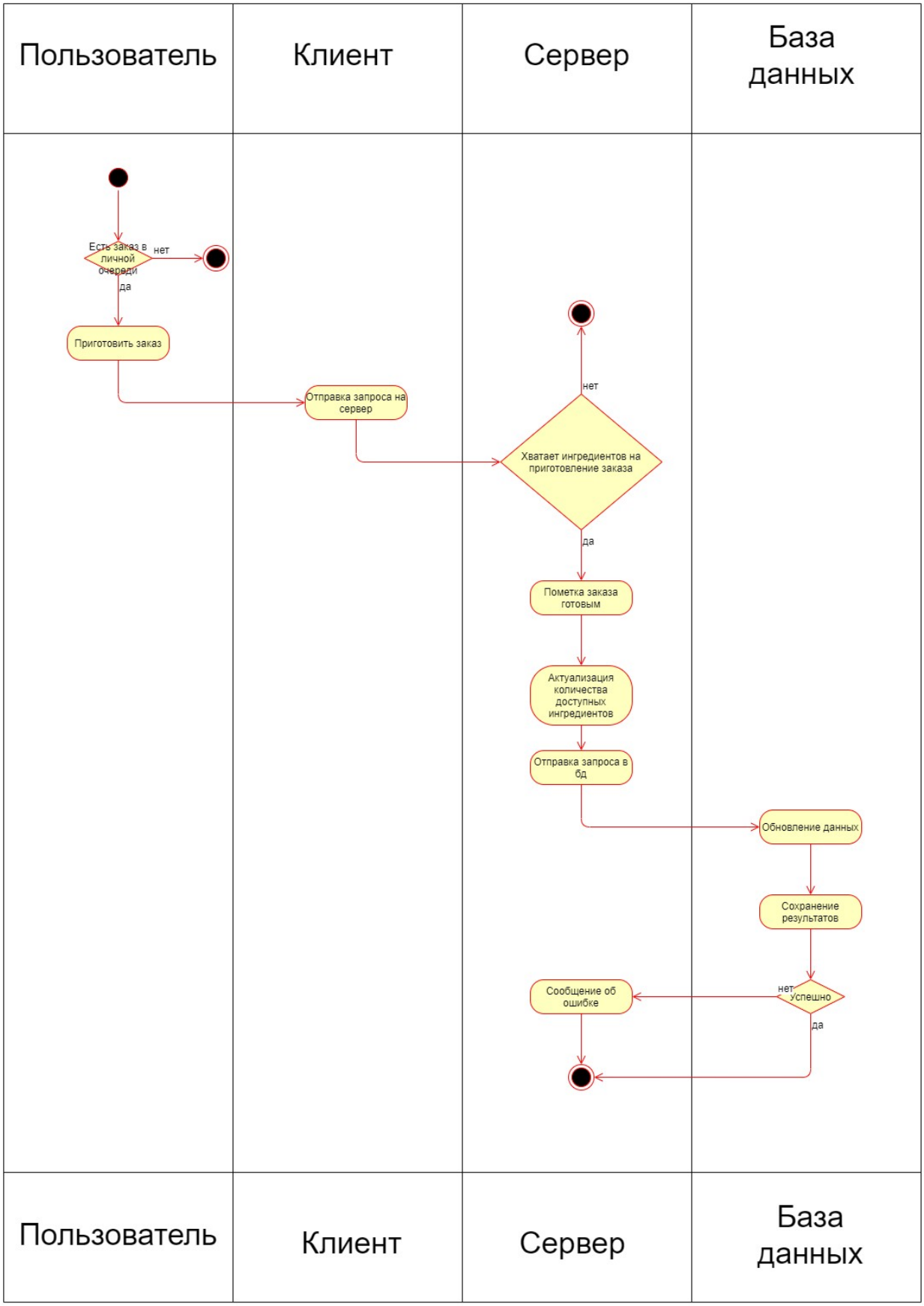


Диаграмма активности для прецедентов «Добавление ингредиентов», «Создание заказа» и «Регистрация сотрудника».

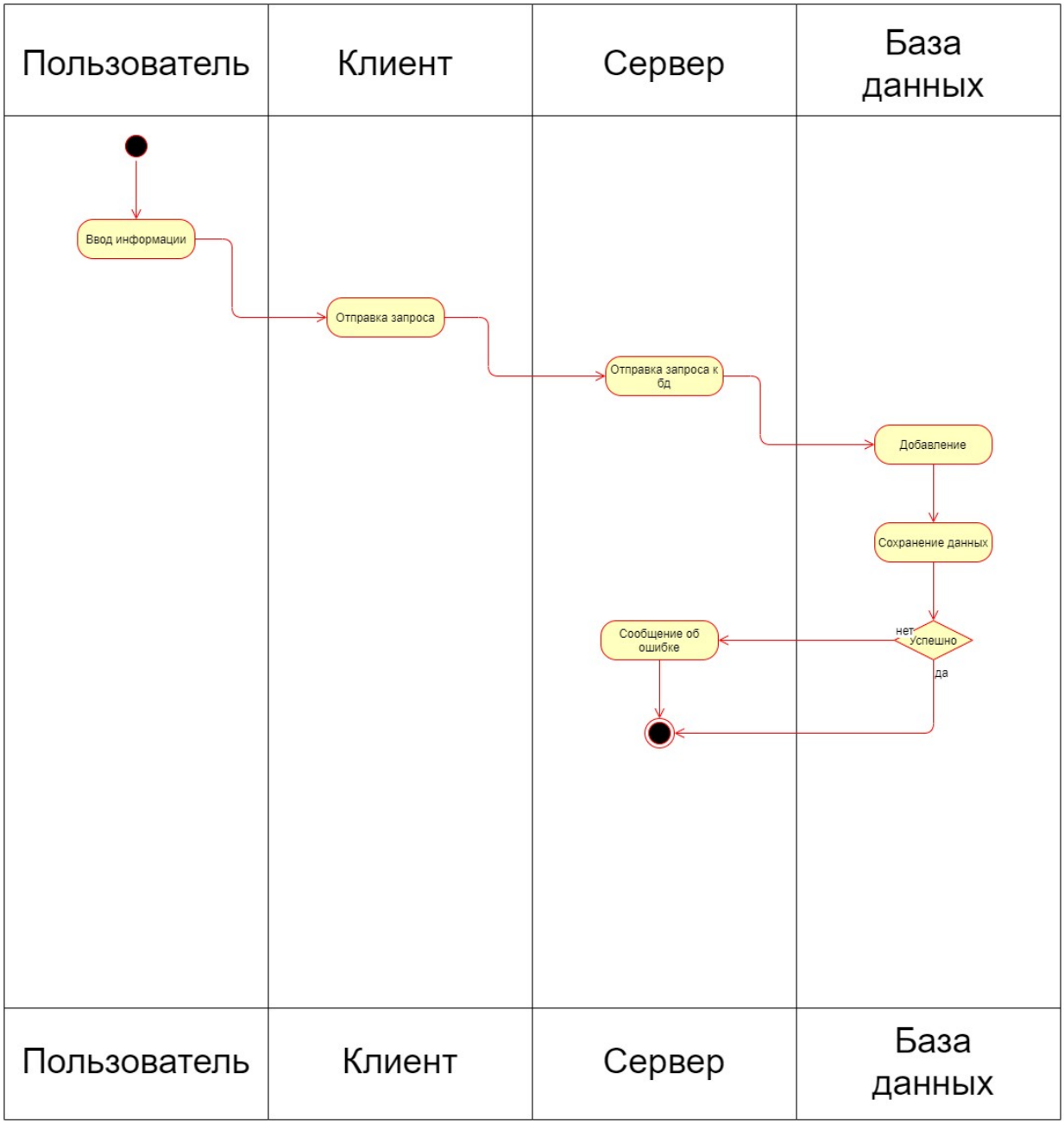


Диаграмма активности для прецедентов «Закупка ингредиентов», «Настройка меню».

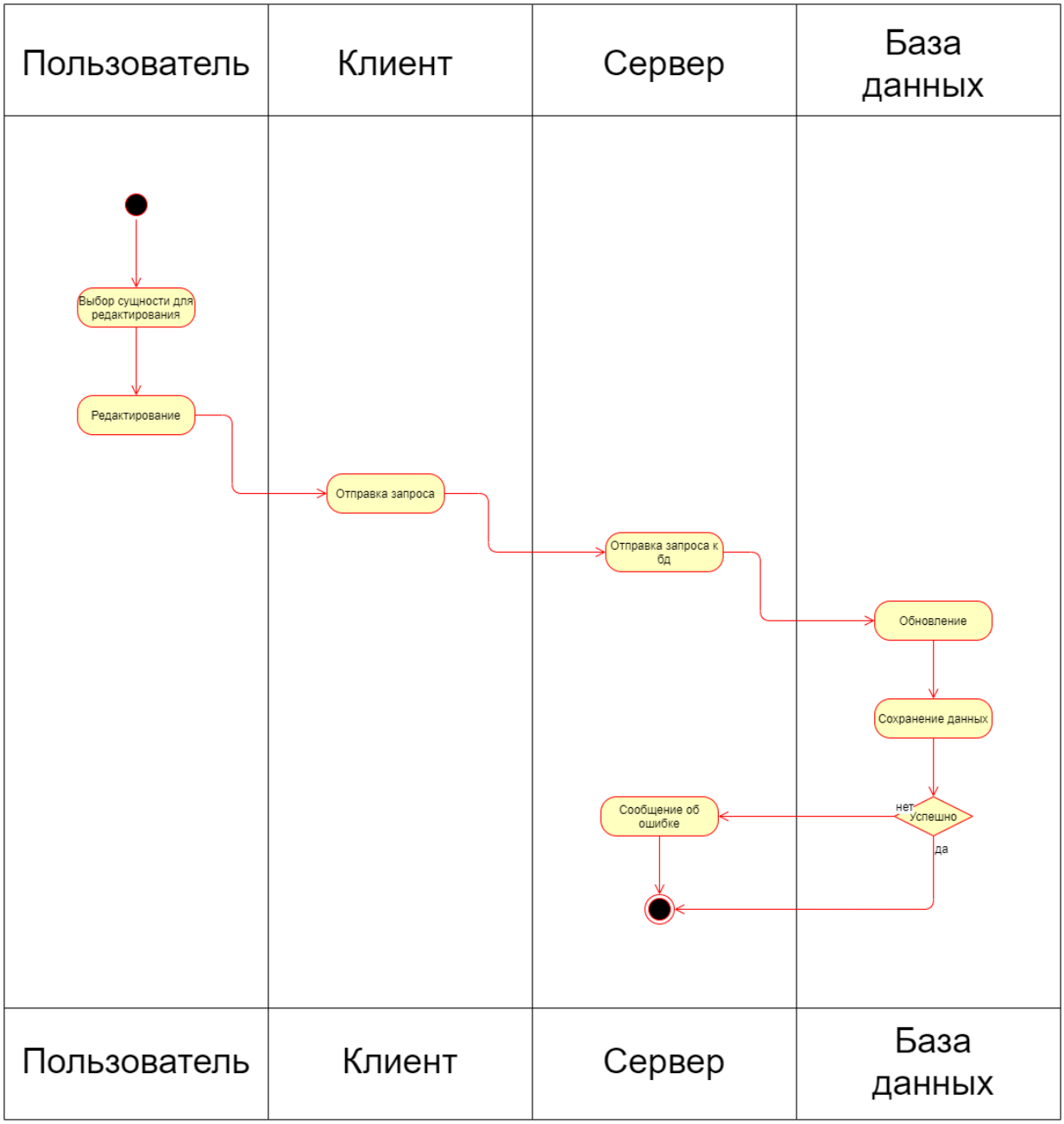
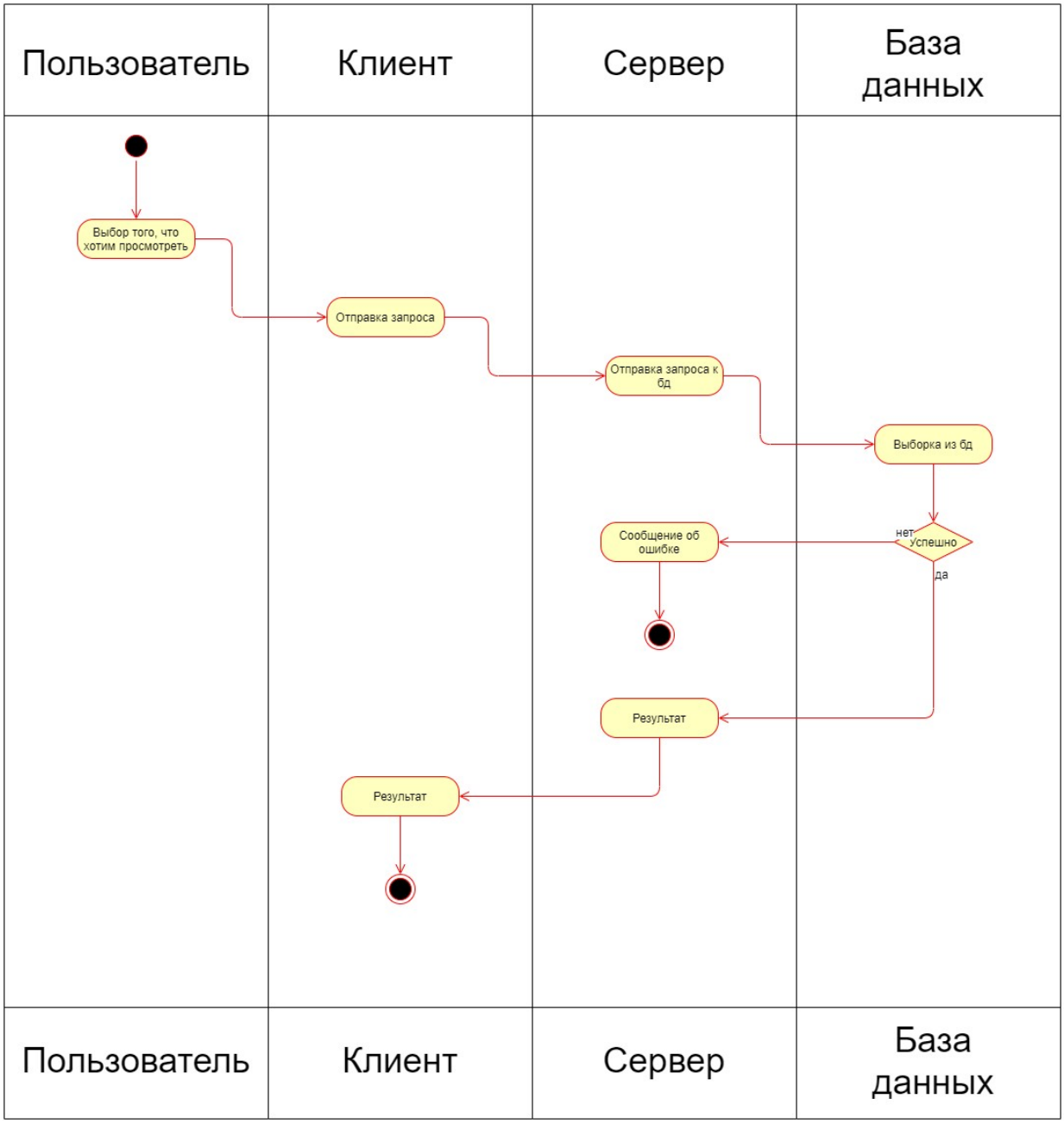


Диаграмма активности для прецедентов «Просмотр статистики», «Просмотр ингредиентов» и «Просмотр сообщений о готовых заказах».


Пользователь
Клиент
Сервер
База данных
Выбор того, что хотим просмотреть
Отправка запроса
Отправка запроса к бд
Выборка из бд
Успешно
нет
да
Сообщение об ошибке
Результат
Результат
Пользователь
Клиент
Сервер
База данных

2. Реализация приложения

Анализ средств реализации

Учитывая требования, которые указаны в Настоящем Техническом Задании, система будет построена на следующих технологиях:

- В качестве API сервера был выбран .NET Core – популярная и прогрессирующая платформа для подобного рода проектов, основанная на базе языка C#.
- В качестве СУБД – MSSQL, так как это классическая связка .NET + MSSQL.
- Entity Framework – самый популярный фреймворк для работы с базой данных в .NET.
- В качестве клиента будет выступать SPA (Single Page Application) приложение на Angular, так как это очень удобный фреймворк на базе языка TypeScript.

Архитектура системы

Приложение будет построено по клиент-серверной архитектуре, с использованием реляционной базы данных.

В качестве клиента используется SPA приложение на Angular, так как генерация одной страницы по тем JSON-данным, которые пришли от сервера, гораздо эффективнее того, если бы сервер возвращал отдельные страницы. Таким образом, мы снимаем нагрузку с сервера, а также это позволяет при необходимости подключать разные API сервисы, причем написаны они могут быть на разных технологиях. К тому же, само слово Angular уже означает больше, чем просто фреймворк. Angular – это front-end фреймворк со своей экосистемой и своими паттернами. Самое популярное – это дробление всего на мелкие компоненты, которые взаимодействуют друг с другом, а также подход реактивного программирования.

В качестве API сервера будет выступать ASP.NET Core, построенный по REST-архитектуре, для взаимодействия с клиентом.

Также в системе используется внедрение зависимостей, то есть Dependency Injection (DI).

В качестве реляционной базы данных выступает Microsoft SQL (MSSQL).

Проектирование системы

Проектирование системы включает в себя:

- Проектирование диаграммы классов
- Проектирование диаграммы объектов
- Проектирование схемы базы данных
- Проектирование диаграммы развертывания

Диаграмма классов

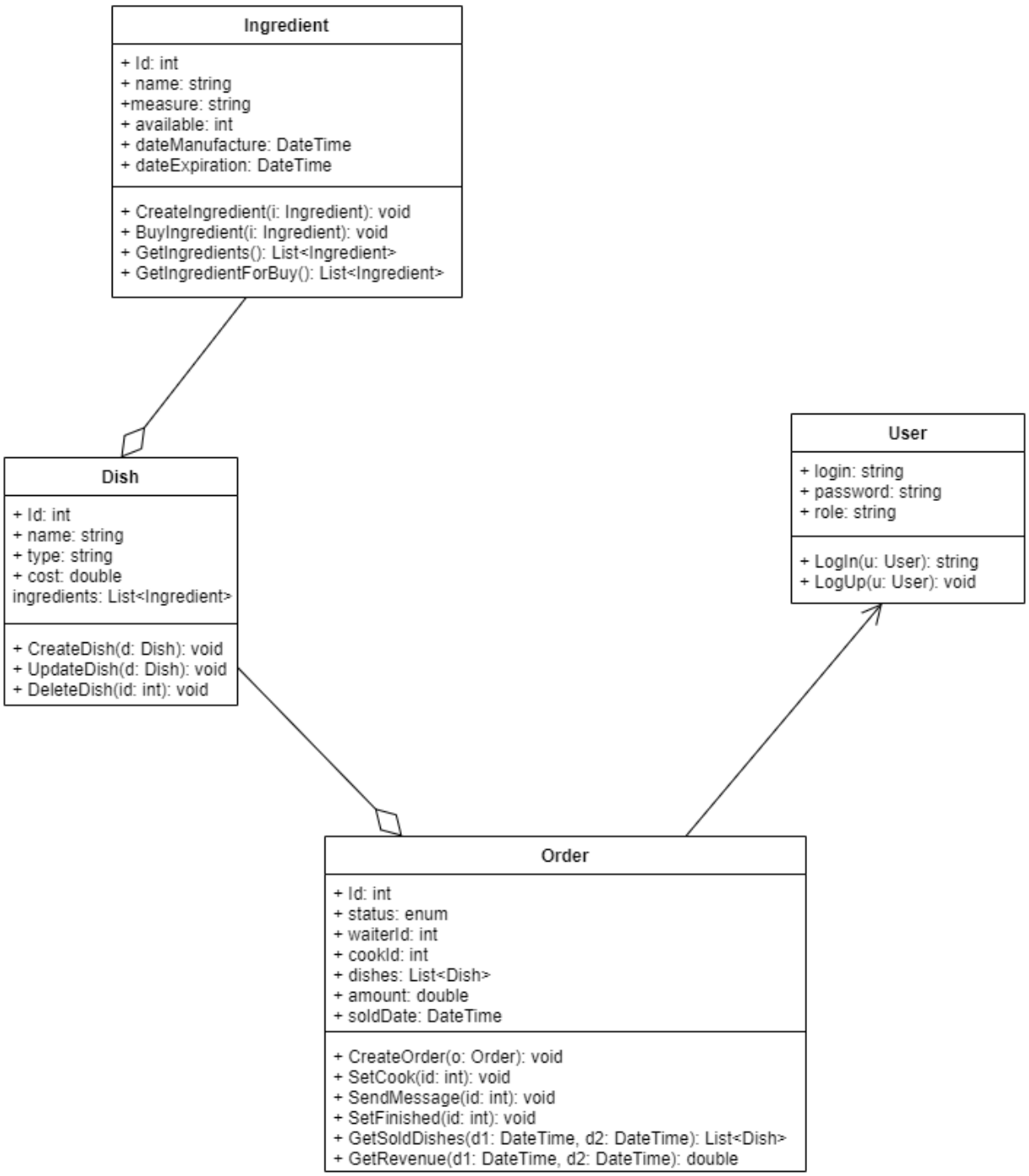


Диаграмма объектов

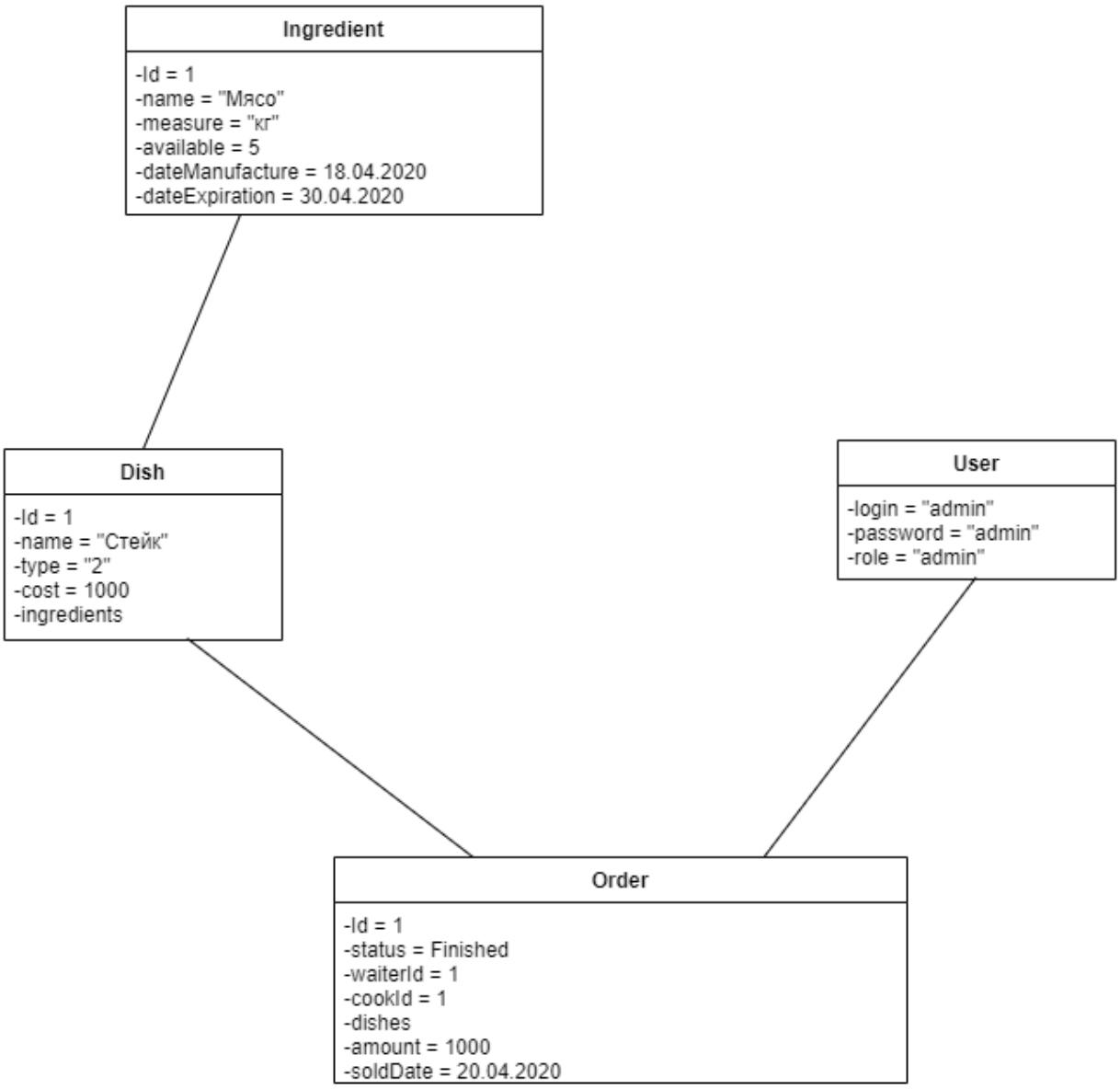


Схема базы данных

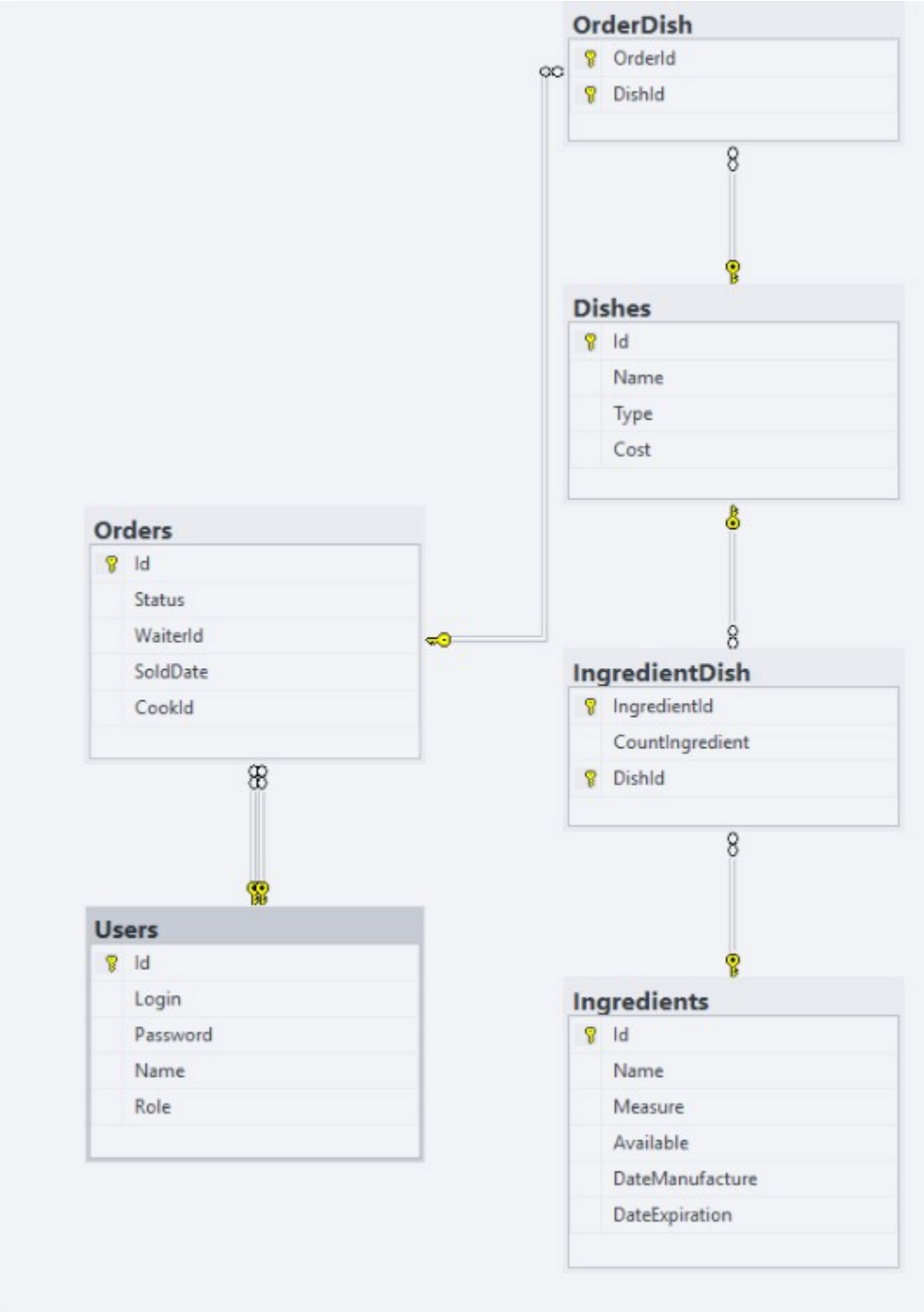
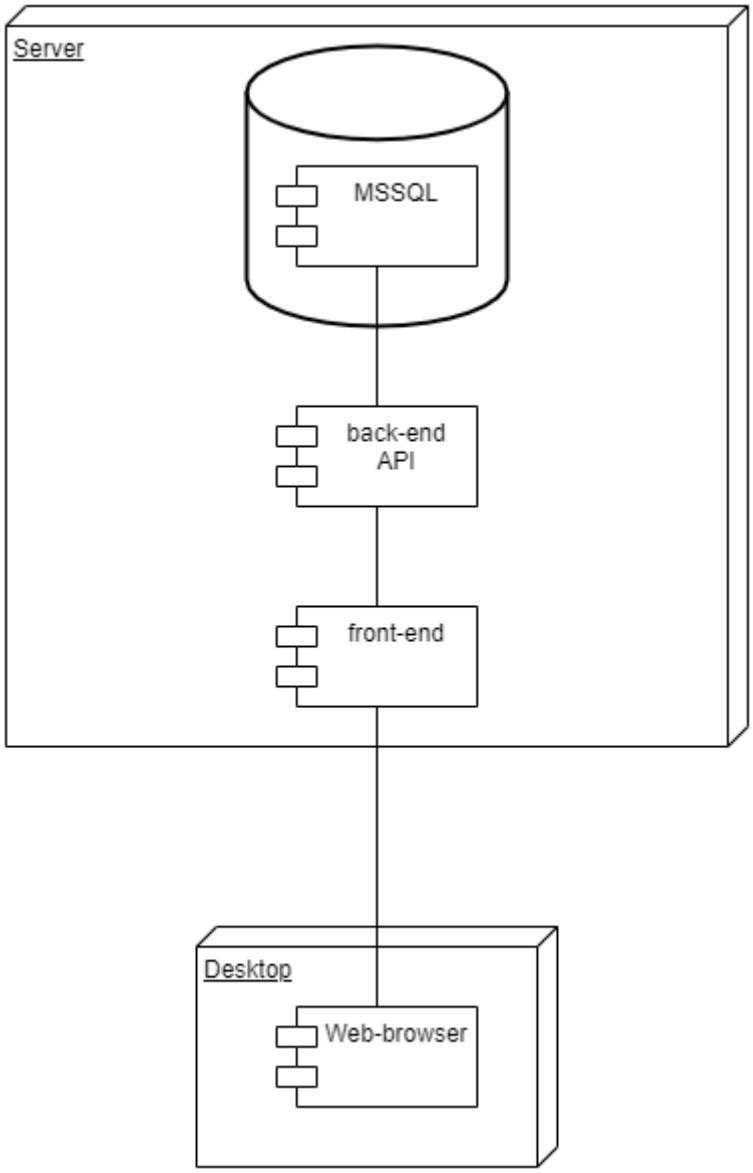


Диаграмма развертывания


Server
MSSQL
back-end
API
front-end
Desktop
Web-browser