2. Проектирование

2.1 Цель раздела

Цель проектирования – это создание фундамента для последующей реализации, тестирования и сопровождения системы. В этом разделе будет определена архитектура системы, структура ее модулей, интерфейсы взаимодействия и методология разработки.

2.2 Архитектура системы

Разрабатываемая система «Система записи на приём к врачу» построена по принципу клиент-серверной двухуровневой архитектуры:

1. Клиентская часть (Frontend): Реализована с использованием HTML5, CSS и JavaScript. Отвечает за отображение интерфейса, валидацию данных, а также за взаимодействие с сервером через REST API.

2. Серверная часть (Backend): Реализована с использованием Node.js и фреймворка Express. Отвечает за обработку запросов, бизнес-логику приложения и работу с базой данных. Для аутентификации и защиты данных используется JWT (JSON Web Tokens).

3. База данных: в качестве базы данных используется SQLite. Осуществляется связь с БД через библиотеку sqlite3.

4. Тип архитектуры: Монолитная с четко выделенными слоями:

– Presentation Layer (интерфейс пользователя);

– Business Logic Layer (обработка данных, валидация, контроль доступа);

– Data Layer (доступ и хранение данных).

Взаимодействие компонентов:

– пользователь → веб-интерфейс → API Express → База данных → ответ пользователю.

2.3 Описание модулей и интерфейсов

Таблица 1 – Описание модулей и интерфейсов

Модуль	Назначение	Входные данные	Выходные данные
AuthModule	Регистрация, аутентификация пользователей | Логин, пароль, данные пользователя	Логин, пароль, данные пользователя	Токен сессии или сообщение об ошибке
ScheduleModule	Просмотр и управление врачами	ID врача, параметры фильтрации	Список врачей с информацией о специализации
BookingModule	Запись на приём, просмотр записей	ID врача, дата, время	Подтверждение записи или сообщение об ошибке
AdminModule	Управление пользователями и врачами	CRUD-запросы	Изменённые записи в БД
DatabaseConnector	Взаимодействие с SQLite БД	SQL-запрос	Результат выборки/изменений

2.4 API спецификация

Пример REST API для ключевых операций:

– POST /api/auth/register

Регистрация нового пользователя.

Request:

```
{
  "username": "ivanov",
  "email": "ivanov@example.com",
  "password": "12345",
  "full_name": "Иванов Иван",
  "birth_date": "1990-01-01",
  "phone": "+7-999-123-45-67"
}
```

Response:

```
{
"success": true,
```

"message": "Пользователь успешно зарегистрирован"

}

– POST /api/appointment

Создание новой записи.

Request:

```
{
  "doctor_id": 3,
  "date": "2025-03-15",
  "time": "14:00"
}
```

Response (успешно):

```
{
  "success": true,
  "appointment": {
    "id": 41,
    "user_id": 123,
    "doctor_id": 3,
    "appointment_date": "2025-03-15",
    "appointment_time": "14:00",
    "status": "scheduled"
  },
  "message": "Запись успешно создана!"
}
```

Response (ошибка):

```
{
"success": false,
"error": "Выбранное время уже занято"
}
```

2.5 UML-диаграммы

2.5.1 Диаграмма вариантов использования (Use Case Diagram)

Диаграмма вариантов использования отображает основные сценарии взаимодействия с системой для двух ролей: Пациента и Администратора. Диаграмма показывает основные шаги, которые выполняются пользователями системы, и их взаимосвязи.

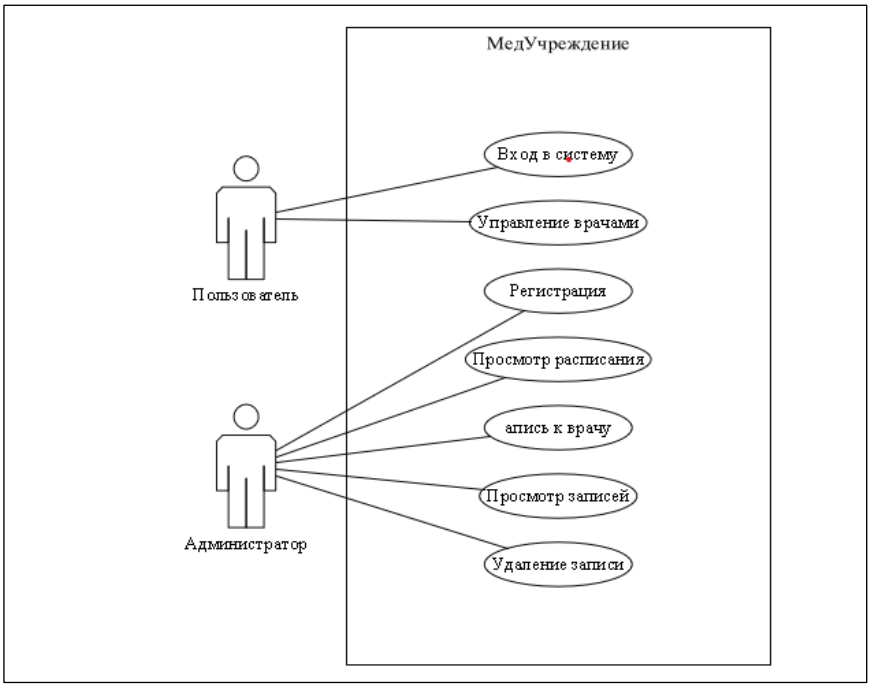

Рисунок 1 - Диаграмма вариантов использования

Пациент выполняет действия по регистрации, авторизации, просмотру расписания, записи и отмене приёма. Администратор управляет расписанием, врачами и пользователями.

2.5.2 Диаграммы последовательности (Sequence Diagram)

В процессе проектирования системы онлайн-записи к врачу особое внимание было уделено созданию диаграмм последовательности для ключевых сценариев взаимодействия пользователей с системой. Мы детально проработали два основных варианта: сценарий пациента и сценарий администратора.

Диаграмма последовательности для главного варианта использования сущности Пациент представлена на рисунке 2.

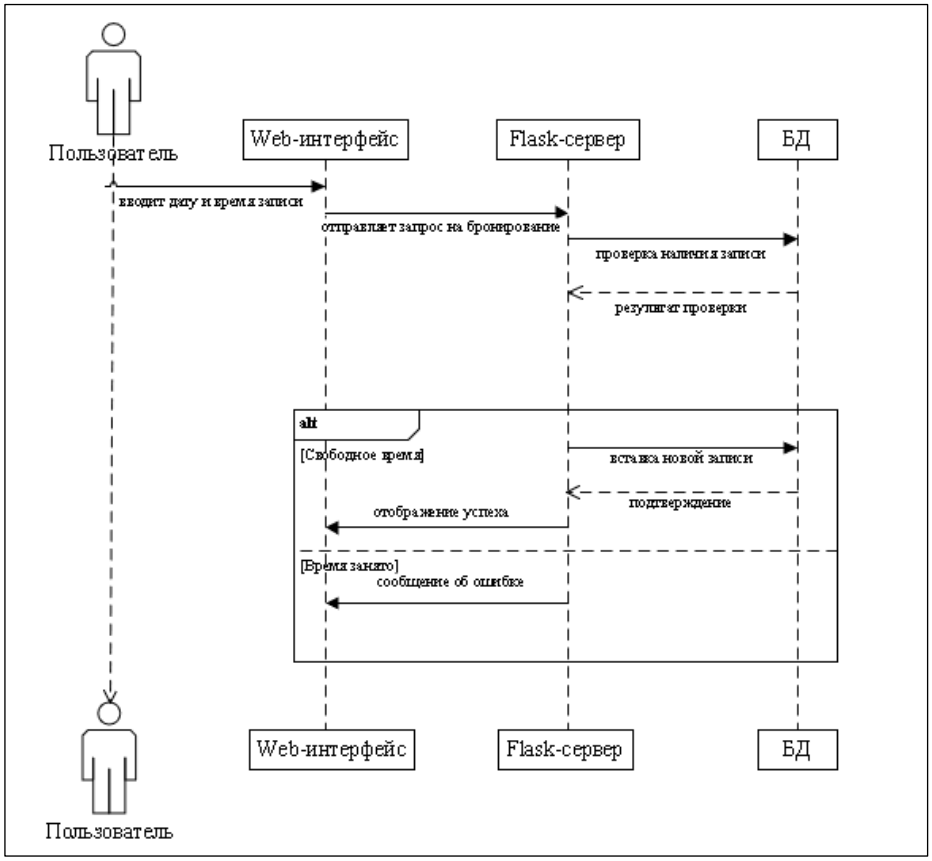

Рисунок 2 - Диаграмма последовательности Пациента

Для сценария пациента диаграмма последовательности показывает полный цикл действий: регистрация или вход в систему, просмотр расписания доступных врачей, выбор времени приёма, подтверждение записи и получение обратной связи о статусе.

Параллельно с этим мы создали диаграмму последовательности для администратора системы, так как его задачи и взаимодействие с системой имеют свои особенности и отличия от пациента. В сценарии администратора особое внимание уделено операциям управления расписанием, подтверждению или отмене приёмов, а также администрированию пользовательских данных и мониторингу состояния системы. Администратор представлена на рисунке 3.

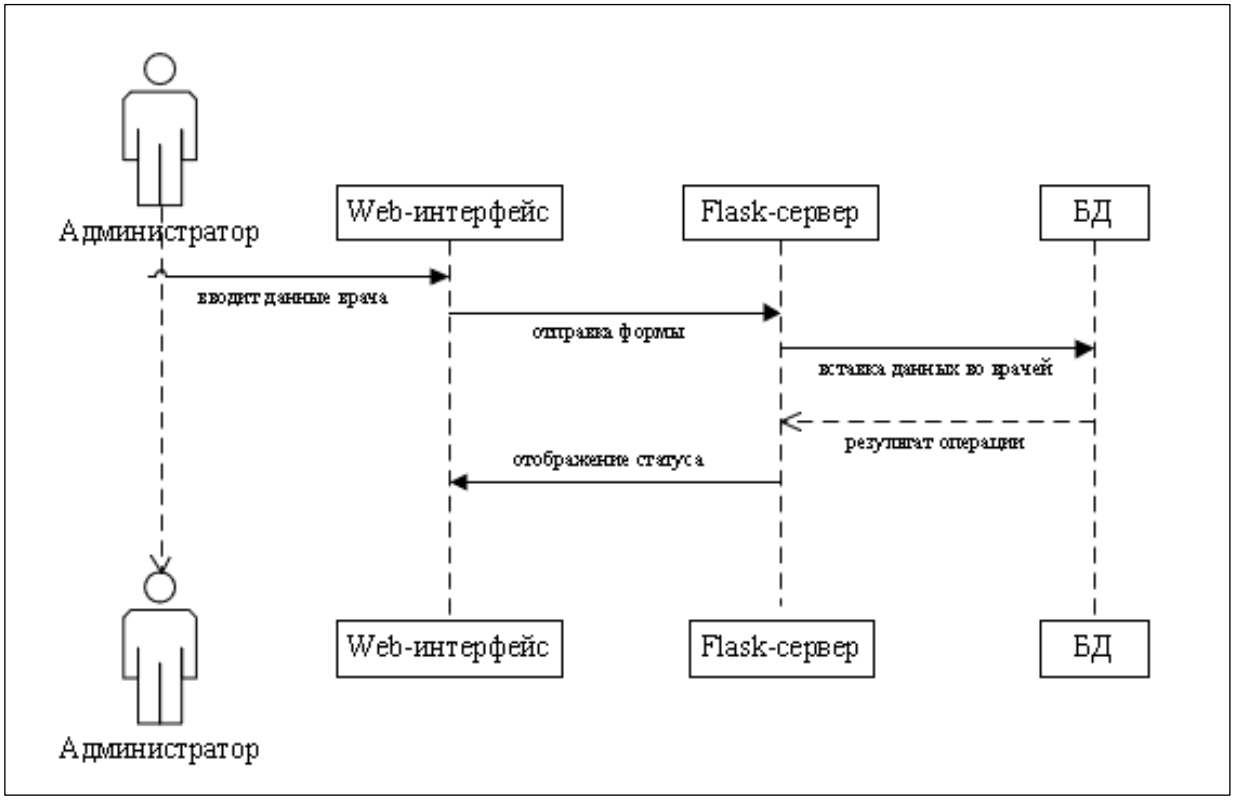

Рисунок 3 - Диаграмма последовательности Администратора

Диаграмма для администратора позволяет чётко представить, как происходит обработка запросов, связанных с поддержкой и управлением системы, а также какие данные и в какой последовательности обрабатываются.

2.5.3 Диаграмма деятельности (Activity Diagram)

С помощью диаграммы деятельности удобно выявлять узлы логики, требующие особого внимания, например, валидацию данных, проверку на занятость слотов или ошибки при доступе к базе данных.

Диаграмма деятельности моделирует поток операций – от входа пользователя в систему до подтверждения или отмены записи. Диаграмма упрощает процесс разработки и тестирования, позволяя заранее предусмотреть все возможные варианты развития событий, минимизируя риски сбоев и повышая общую стабильность системы. Диаграмма представлена на Рисунке 4.

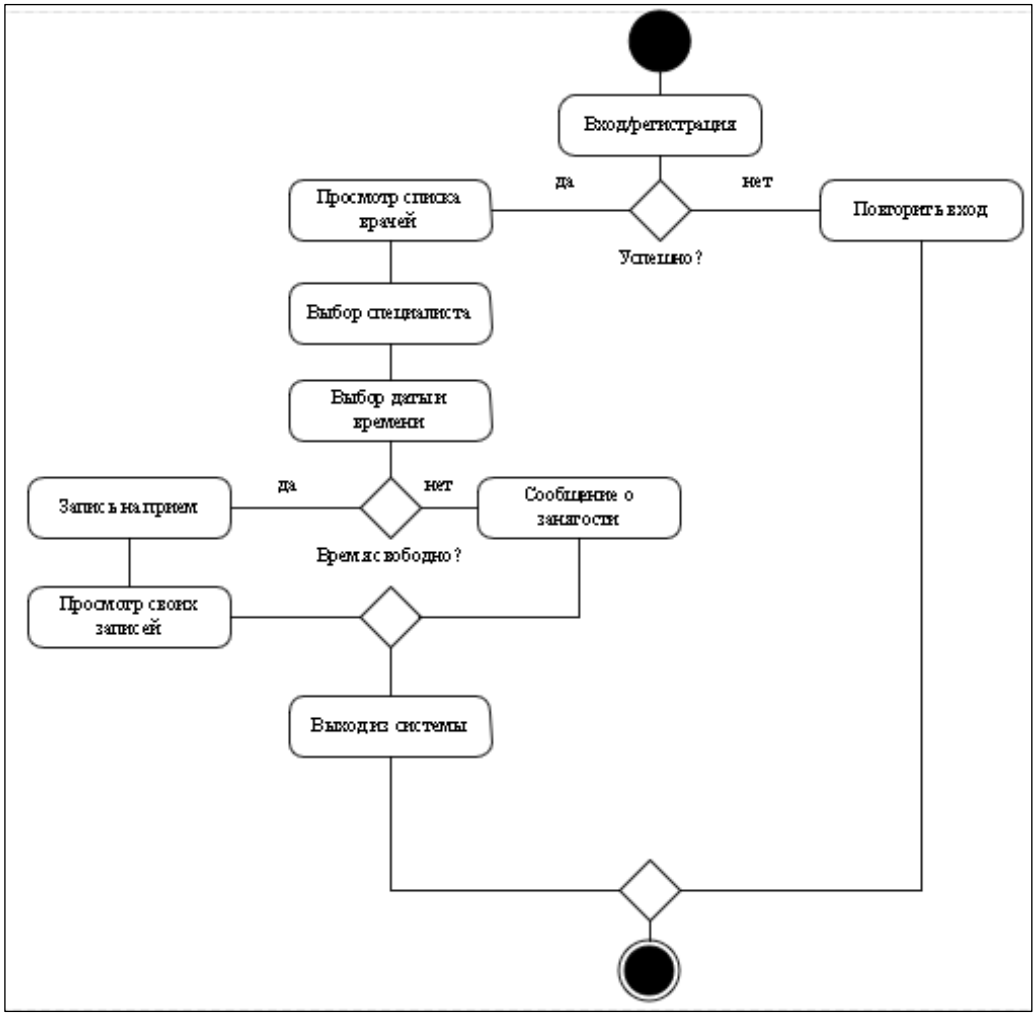

Рисунок 4 - Диаграмма деятельности

Использование этой схемы упрощает взаимодействие между разработчиком и дизайнером интерфейсов, поскольку все возможные действия заранее визуализированы и легко адаптируются под реальные условия.

2.6 ER-диаграмма базы данных

Завершающим этапом архитектурного проектирования является построение ER-диаграммы (Entity-Relationship diagram), которая описывает структуру базы данных. Данная диаграмма играет критическую роль в формировании хранилища информации, так как определяет, какие сущности существуют в системе, какие атрибуты они содержат и как между ними устанавливаются связи. ER-диаграмма базы данных изображена на рисунке 5.

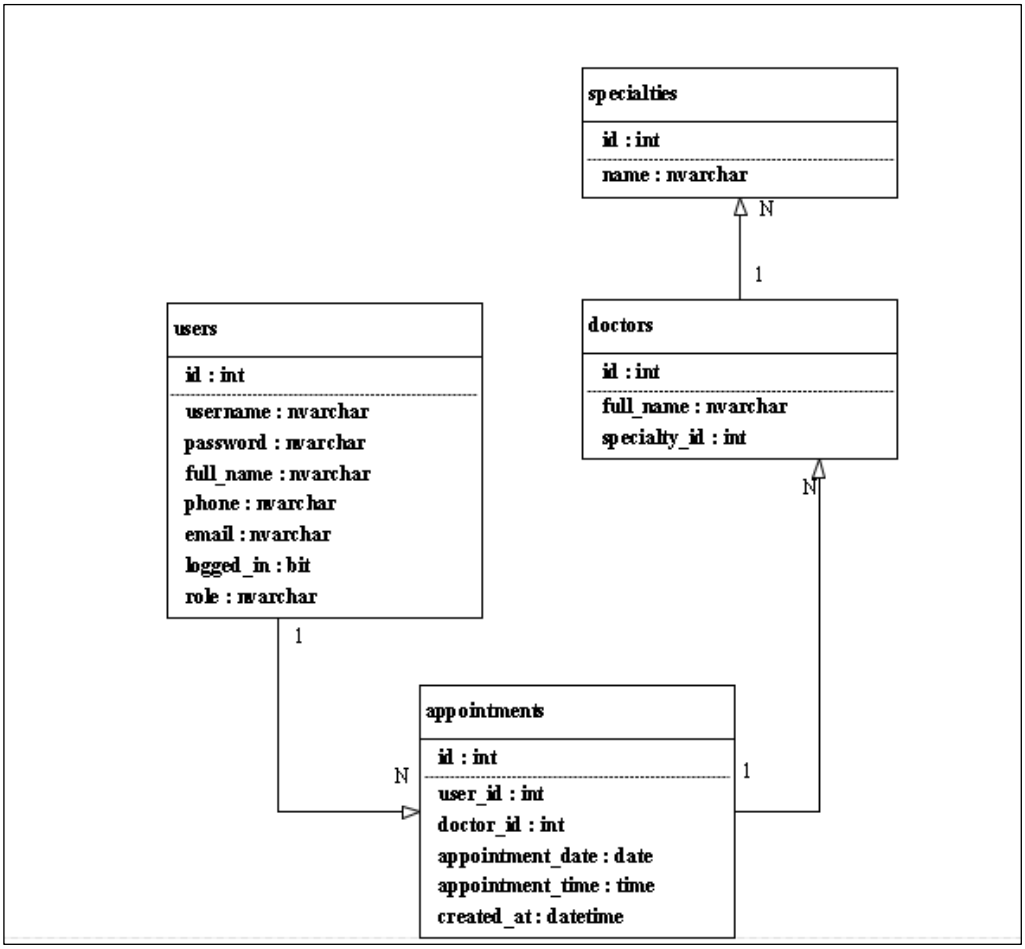

Рисунок 5 - ER-Диаграмма

ER-диаграмма описывает сущности, которые будут использованы в базе данных, а также связи между ними.

– User: id, username, email, password_hash, full_name. birth_date, phone, role, created_at;

– Doctor: id, name, specialization, created_at;

– Schedule: id, doctor_id, date, time_slot, is_available, created_at;

– Appointment: id, user_id, doctor_id, date, time_slot, status, created_ad.

Связи:

– один Doctor может иметь несколько Appointment;

– один User может иметь несколько Appointment;

– один Appointment связан с одним Doctor и одним User.

2.7 План тестируемости

Для обеспечения корректности работы системы, будут проведены различные уровни тестирования.

Таблица 2 – План тестируемости

Уровень	Модуль	Тип тестов
Unit	AuthModule, BookingModule	Тесты логики и валидации
Integration	API и база данных	Проверка запросов и ответов
System	Вся система через интерфейс	Тестирование пользовательских сценариев
Regression	CI-пайплайн	Проверка стабильности после обновлений

Цель – достичь покрытия тестами не менее 75% ключевых модулей.

2.8 Требования к безопасности и резервированию

– Все пароли хранятся в зашифрованном виде с использованием bcrypt.

– Все операции записи требуют аутентификации через JWT.

– Ежедневное резервное копирование базы данных.

– Административные функции доступны только пользователям с соответствующими правами.