|  |  |  |  |
| --- | --- | --- | --- |
| **ПРИЛОЖЕНИЕ Б** | | | |
|  | |  | |
|  |  |  |  |
|  |  |  |  |
|  | | | |
| **Технический проект**  веб-приложения “NeuroKanban”  на 14 листах  2023 | | | |
|  | | | |

**Содержание**

[1 Описание проекта 27](#_Toc150200495)

[2 Основные функциональные возможности 28](#_Toc150200496)

[3 Уточнение структуры данных 30](#_Toc150200497)

[4 Формы представления данных 33](#_Toc150200498)

[5 Архитектура системы 34](#_Toc150200499)

[6 Разработка алгоритма решения задачи 35](#_Toc150200500)

[7 Определение языка, структуры программы и требований к техническим средствам 36](#_Toc150200501)

[8 Требования к техническим средствам 38](#_Toc150200502)

# Описание проекта

Название проекта: Интернет-сервис NeuroKanban для организации работы проектной группы над конкретными проектами посредством парадигмы канбан.

Цель проекта: разработка аналога популярных Интернет-сервисов, ушедших с Российского рынка, а также соответствие требованиям FOSS и работа внутри организации.

# Основные функциональные возможности

NeuroKanban должен обеспечивать возможности:

* авторизация в системе посредством логина и пароля;
* наличие ролей руководителя проекта и участника;
* навигация на сайте посредством пользовательского интерфейса;
* создание и редактирование проектов в системе;
* приглашение пользователей в проект от руководителей проектов;
* удаление пользователей из проекта;
* создание и редактирование «досок» внутри проектов;
* создание и редактирование групп задач (столбцов) на «досках»;
* создание, редактирование, перемещение задач между группами на «досках», а также установление различных характеристик задач, таких как трудоемкость (крайняя дата выполнения, уже затраченное время исполнения);
* обеспечение напоминаний о заканчивающемся сроке для выполнения задач;
* информирование руководителей проекта о статусе выполнения задач.

Бизнес-логика и ее декомпозиция представлены на рисунках 2.1 и 2.2.

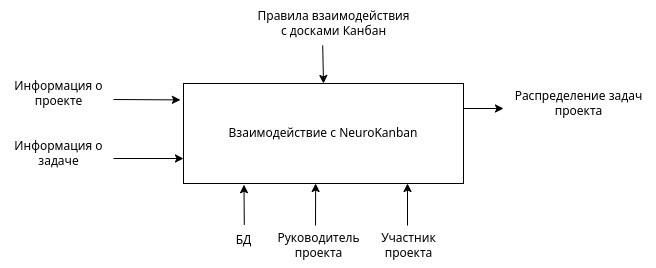


Рисунок 2.1 - Функциональная диаграмма IDEF0

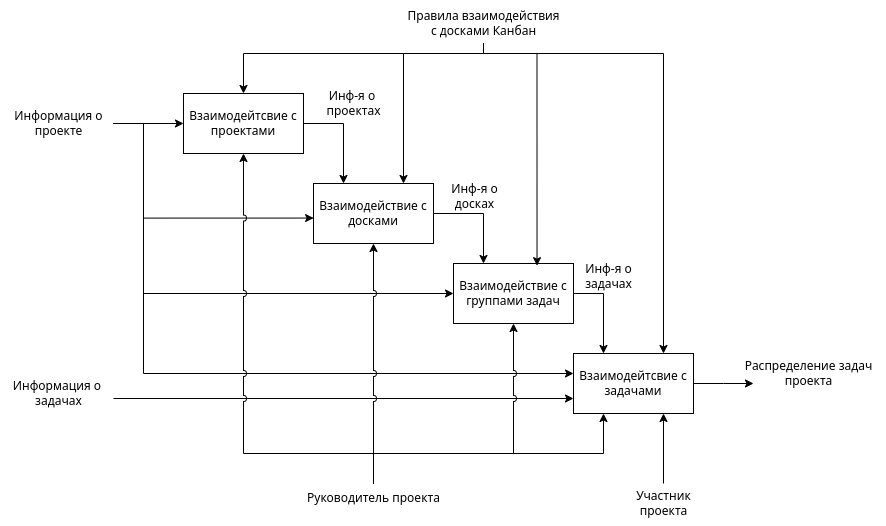


Рисунок 2.2 - декомпозиция IDEF0

# Уточнение структуры данных

Потоки данных в системе представлены представлены на рисунке 3.1 в виде DFD модели, предназначенных для описания потоков данных, происходящих в информационной системе NeuroKanban, показывающая, какие данные и процессы происходят в системе.

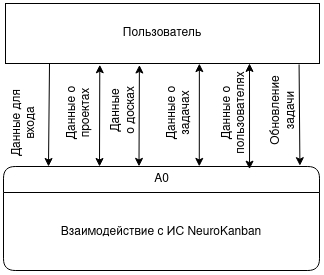


Рисунок 3.1 - нулевая декомпозиция DFD

При взаимодействии с системой пользователь в первую очередь проходит авторизацию, затем может взаимодействовать с проектами, внутри которых возможно взаимодействие с привязанными к ним досками, где, в свою очередь, возможно взаимодействие с группами задач, привязанных к доскам, где также можно взаимодействовать с задачами, а также изменять их статус. Также возможно добавлять в проект новых пользователей.

Декомпозиция DFD представлена на рисунке 3.2.

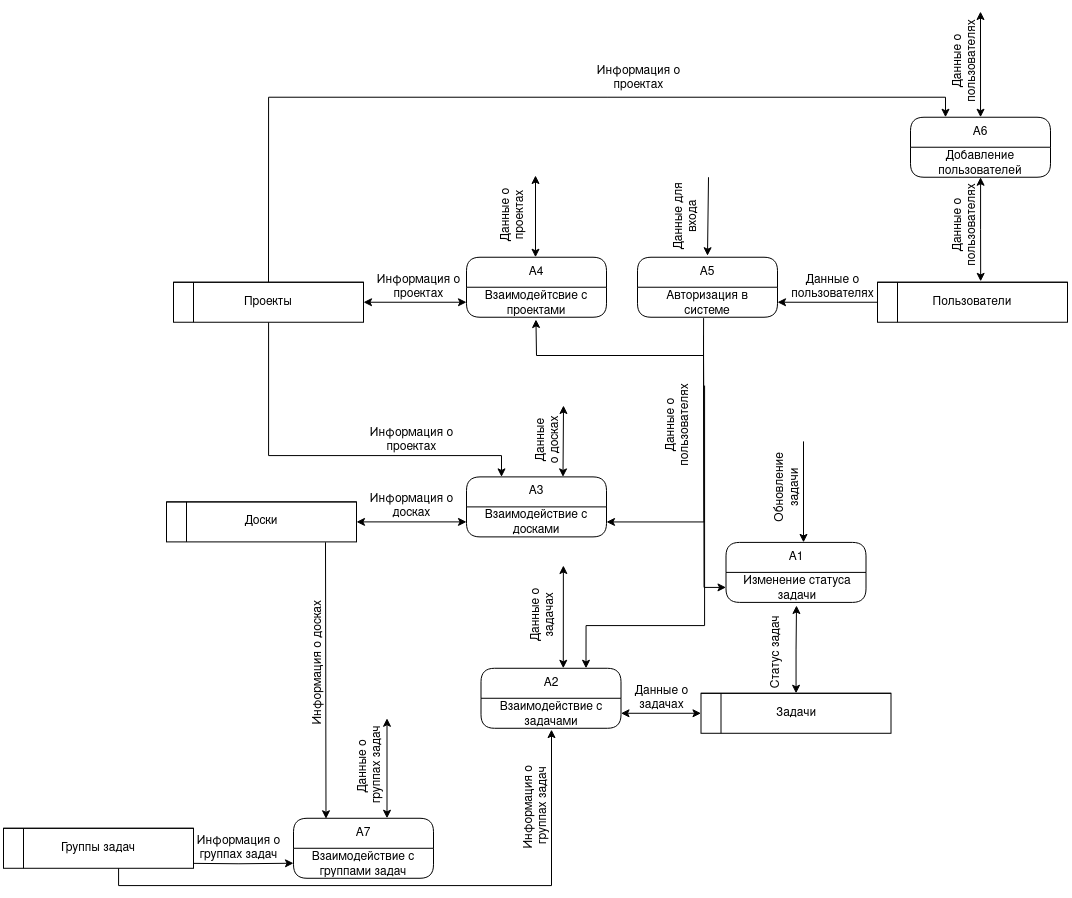


Рисунок 3.2 - первая декомпозиция DFD

Используемые структуры данных представлены посредством ER-диаграммы, изображенной на рисунке 3.3.

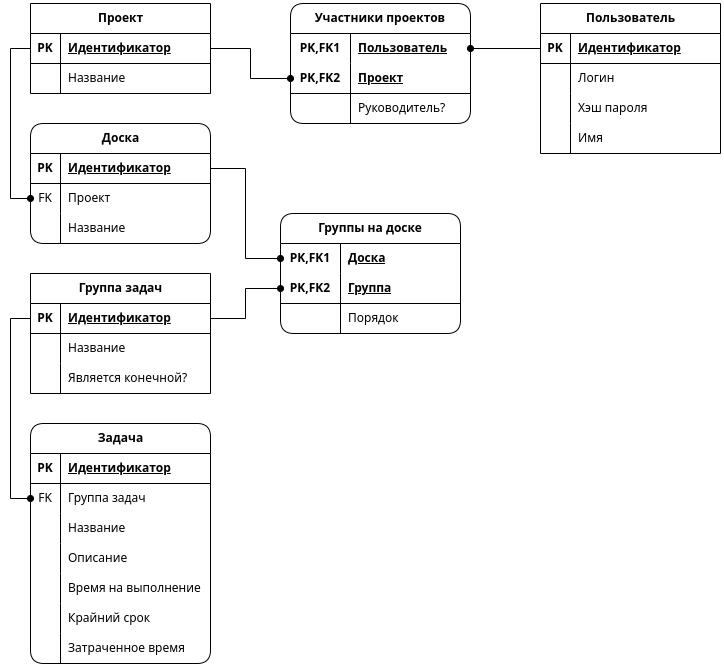


Рисунок 3.3 - концептуальная ER-диаграмма

# Формы представления данных

При попадании в систему, пользователь в первую очередь выполняет вход в систему посредством ввода логина и пароля в соответствующих полях ввода.

После входа пользователь попадает на экран с проектами, где может быть выбран существующий проект, либо создан новый.

После выбора проекта, пользователь попадет на экран «досок» проекта, где можно выбрать существующую «доску», либо создать ее, а также пригласить участника в проект или удалить участника. После выбора «доски» проекта, происходит переход на экран выбранной «доски», где можно создать новую группу задач, создать новую задачу в любой из групп, взаимодействовать с созданными задачами.

# Архитектура системы

В данной системе используется двухзвенная архитектура “клиент-сервер”. Подразумевается взаимодействие множества клиентов с сервером посредством “тонкого” клиента - веб-сайта у множества клиентов, подключаемых к одному серверу обработки данных с СУБД и БД. Схема архитектуры представлена на рисунке 5.1.

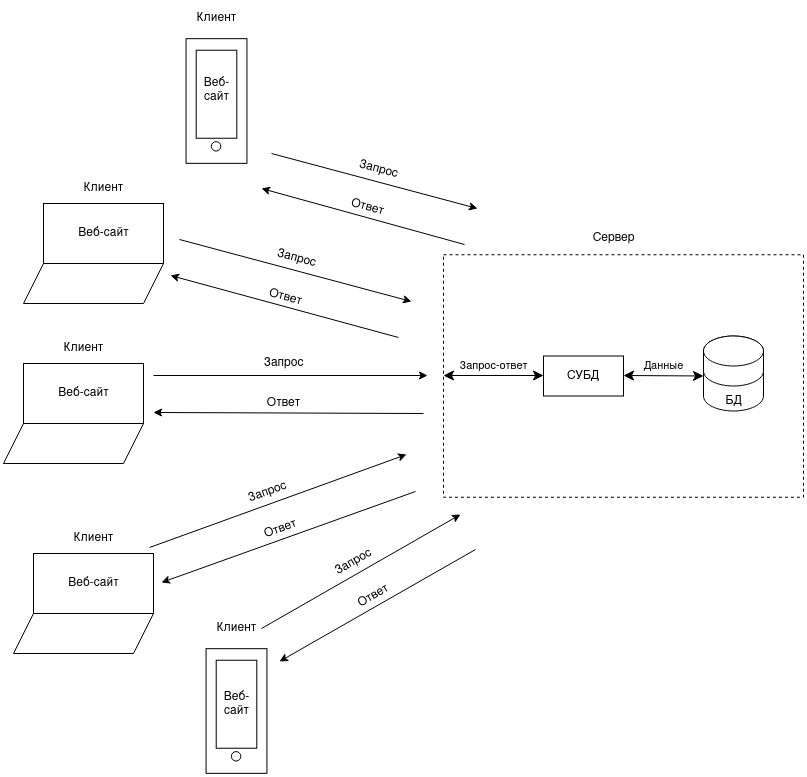


Рисунок 5.1 - схема архитектуры системы

# Разработка алгоритма решения задачи

В системе возможны три роли: администратор, руководитель проекта, участник проекта.

Форма входа проверяет наличие пользователя в базе данных и в случае неверно введенного логина или пароля сообщает об отсутствии этого пользователя в системе.

Формы для ввода информации осуществляются HTTP запросами с клиентской части на серверную, которая затем обрабатывает полученную информацию, записывает ее в базу данных при помощи средств ORM-инструментария библиотеки Prisma. Вывод информации осуществляется HTTP запросом с клиентской части на серверную часть, что затем делает выборку по базам данных, исходя из запроса, также при помощи Prisma.

При попытке редактирования или удаления чего-либо в системе происходит проверка наличия прав на выполнение данных действий сверкой пользователя с руководителем проекта в базе данных, а затем уже выполнение самих этих действий в случае успеха проверки, либо выдача ошибки в случае неудачи.

При добавлении задачи есть ограничение на ввод даты и времени окончания задачи. Данное ограничение не дает ввести дату и время в некорректном формате. Проверка должна производиться как на клиентской части, так и на серверной.

Для произведения изменений на клиентской части в режиме реального времени держится WebSocket (в данном случае Socket.io) соединение между клиентской и серверной частью, отправляя новую информацию от сервера к клиентам, когда происходят изменения в данных системы другими пользователями. Таким образом, это позволяет пользователям видеть, как задачи меняют свои группы, редактируются названия проектов, досок, групп задач, свойства задач и т.д. в режиме реального времени, без необходимости перезагрузки веб-страницы.

Каждую минуту происходит выборка по задачам при помощи Prisma и происходит отправка пользователям, посредством массовой рассылки через Gmail API, задач, до крайнего срока которых осталось 10 минут.

При добавлении новых участников в проект в первую очередь происходит проверка ввода, чтобы ввод соответствовал маске электронной почты. Маска представляет собой регулярное выражение вида “^((?!\.)[\w\-\_.]\*[^.])(@\w+)(\.\w+(\.\w+)?[^.\W])$”. Проверка происходит и на клиентской части, и на серверной. Затем им отправляться на электронную почту автоматически сгенерированные системой пароль, если они ранее не были зарегистрированы в системе, в ином случае им будет отправлено на электронную почту уведомление о добавлении в проект.

# Определение языка, структуры программы и требований к техническим средствам

Выбранным архитектурным решением является использование архитектуры “Клиент-Сервер” с использованием СУБД. Серверная часть обрабатывает запросы от клиентов, работает с базой данных и возвращает результаты запросов клиентам. Клиентская часть делает запросы к серверной части, получает ответ от сервера и выводит его в нужном виде на интерфейсе сайта.

Данное решение подразумевает, что клиентов, что одновременно взаимодействуют с сервером будет множество, в то время как взаимодействовать с СУБД будет лишь один бэкенд сервер (серверная часть веб-сайта).

* Клиентская часть (Фронтенд):

язык программирования TypeScript. Выбран в связи с наличием статической типизации и большим набором библиотек для реализации клиентской части веб-приложений;

библиотека Socket.io. Необходима для загрузки обновлений с сервера, не используя перезагрузку страницы;

фреймворк React. Выбор пал на него, так как он имеет виртуализацию DOM, что позволяет делать веб-страницы без загрузок новых страниц, а также компонентный подход к реализации;

веб-сервер nginx. Выбран так как является простым в развертывании, а также благодаря простой поддержке как обычных, так и WebSocket соединений.

* Серверная часть (Бэкенд):

язык программирования TypeScript;

фреймворк express.js. Был выбран, так как является гибким, быстрым в маршрутизации, имеющим хорошую поддержку REST API;

библиотека Socket.io. Необходима для поддержки WebSocket соединения с клиентом;

СУБД SQLite. Выбрана так как приложение не имеет необходимости в хранении и обработке больших массивов данных, а также нет необходимости доступа к базе данных с разных серверов компаний серверной части приложения, так как для каждой компании будет реализована отдельная серверная часть.

Как серверная, так и клиентская часть развертываются за счет Docker-контейнеров и инструмента Docker Compose, позволяющего разворачивать приложения с несколькими контейнерами. Данная система развертывания выбрана так как благодаря ней появляется возможность развертывания на многих конфигурациях серверов, использующих различные операционные системы, так как работа происходит при помощи контейнеризации.

# Требования к техническим средствам

Серверная часть:

* Операционная система: GNU/Linux Ubuntu 22.04,
* ОЗУ: 1 ГБ либо более;
* Жесткий диск: 5 ГБ либо более;
* Быстрое и надежное сетевое подключение с высокой скоростью передачи данных.
* Клиентская часть:
* Устройство, поддерживающее браузер (смартфон, ПК);
* Доступ к интернету для взаимодействия с веб-сайтом.