

Project: ToDoList

Autor: Agashirinov K.A.

March 2023

Saint-Petersburg

## **Содержание**

- |                                          |       |
|------------------------------------------|-------|
| 1. Функциональная спецификация.....      | 3     |
| 2. Руководство пользователя.....         | 3-11  |
| 3. Архитектура программы.....            | 11-12 |
| 4. Описание наиболее важных классов..... | 12-15 |

## **1. Функциональная спецификация**

Данный проект представляет собой список дел, который позволяет создавать заметки. Каждому пользователю, чтобы пользоваться сервисом необходимо пройти регистрацию на сайте.

Пользователь имеет возможность создавать заметки. При добавлении заметки автоматически появляется дата создания заметки, также пользователь указывает название, описание, срок выполнения. Редактируя заметку можно изменить название, описание, срок выполнения. Также есть возможности писать несколько комментариев к заметке, удалять, делать поиск по названию. Пользователь также может увидеть свои данные в настройках.

## **2. Руководство пользователя**

При переходе на сайт пользователя встречает главная страница (Рис. 1). В правом верхнем углу две ссылки для входа пользователя и его регистрации на сайте.

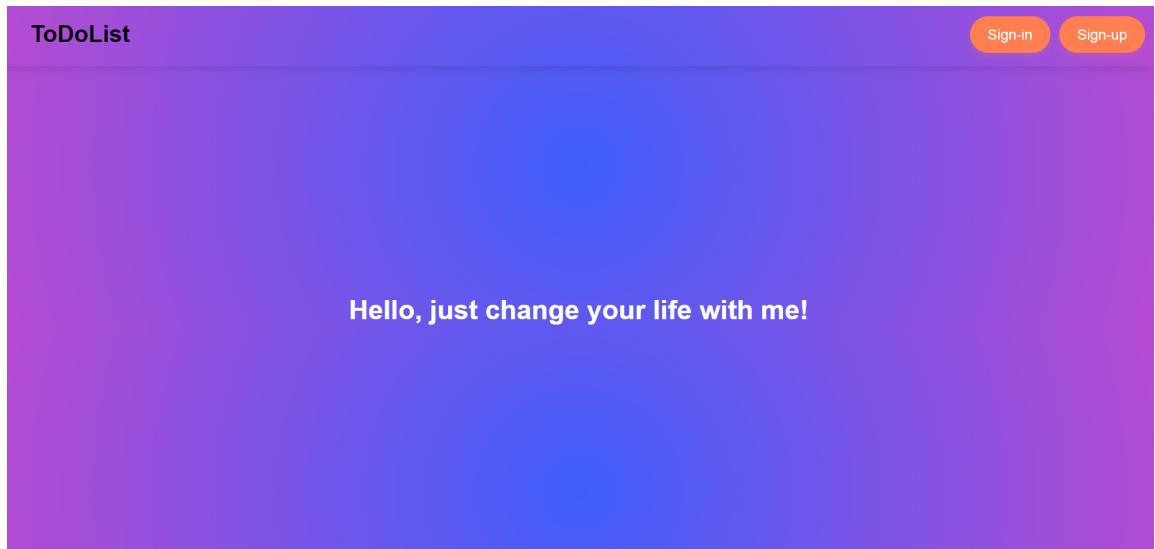


Рис. 1 : главная страница.

При нажатии на кнопку 'Sing-in' пользователь попадает на страницу с формой для входа, которая представлена на (Рис. 2). Если пользователь, у которого нет учетной записи попытается войти в систему, то будет ошибка. Сначала ему нужно пройти регистрацию. При вводе неправильного пароля или логина будет предоставлена страница для регистрации. Также есть возможность перейти на форму регистрации.

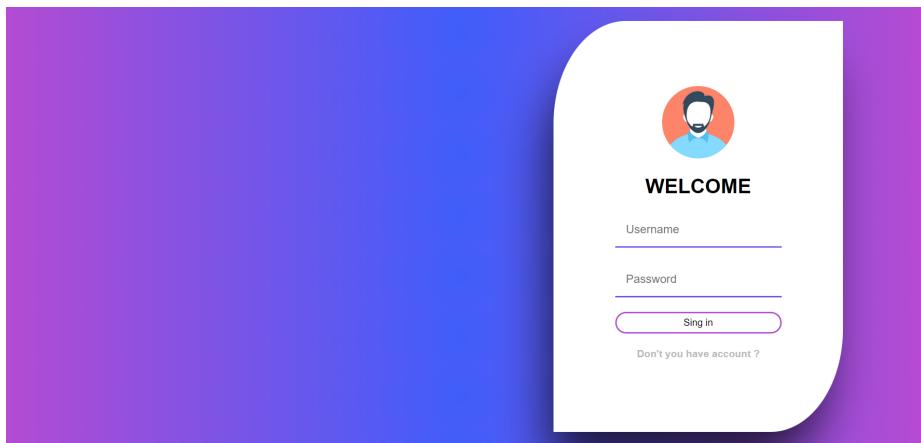


Рис. 2 : вход пользователя.

При нажатии на кнопку 'Зарегистрироваться' пользователь попадает на страницу с формой для регистрации, которая представлена на (Рис. 3). При вводе почты в некорректном формате и уже существующем имени будут выдаваться ошибки. Также есть возможность перейти на форму авторизации.

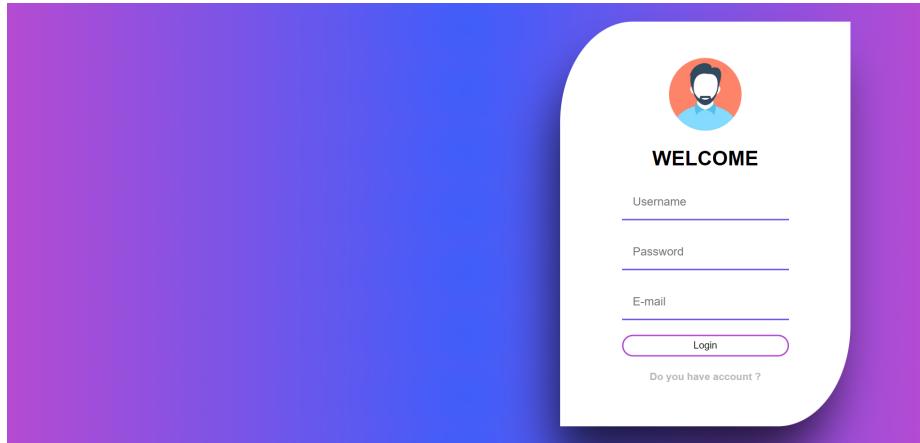


Рис. 3 : регистрация пользователя.

После входа пользователя в систему его встречает первая заметка от системы и также у него появляется доступ ко всем возможностям сервиса (Рис. 4).

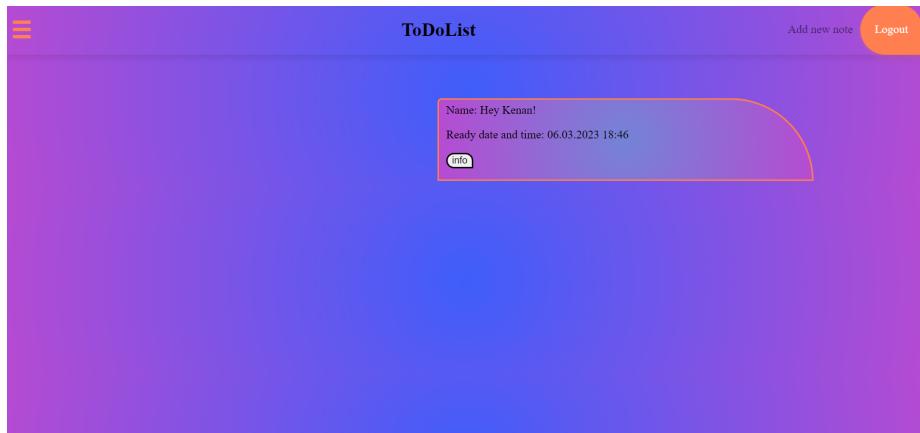


Рис. 4 : начальная страница пользователя.

Чтобы создать новую заметку нужно нажать на кнопку 'Add new note'. При нажатии на кнопку 'Add new note' пользователь переходит на страницу для создания новой заметки (Рис. 5).

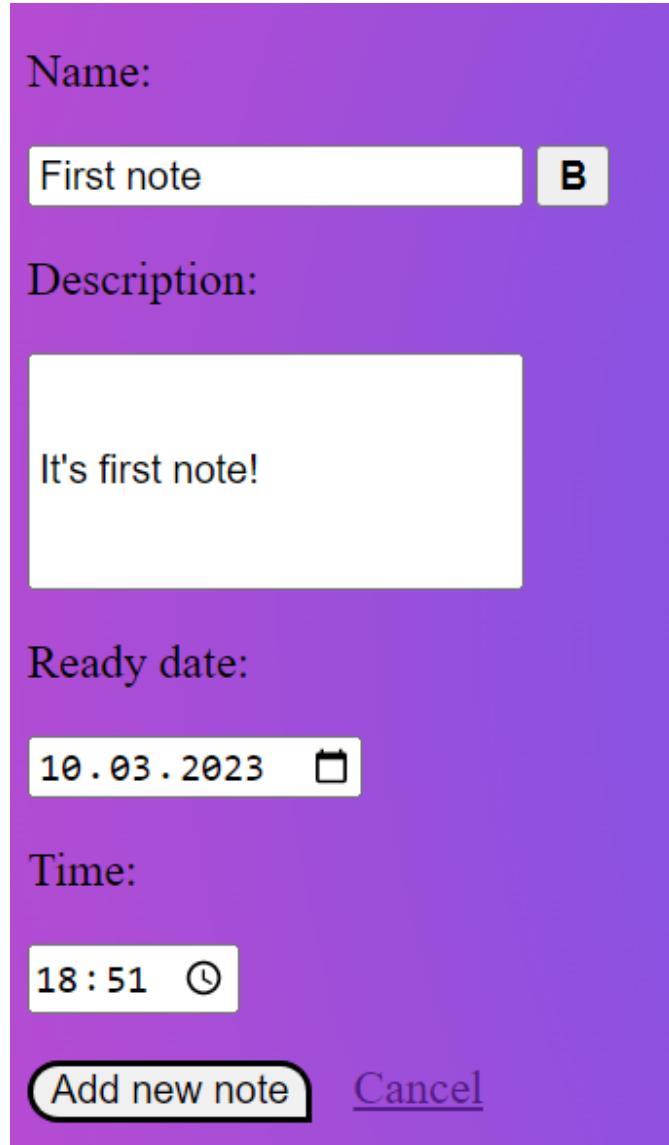


Рис. 5 : добавление новой заметки.

После создания заметок (Рис. 6), пользователь может открыть заметку для подробной информации нажав на кнопку 'info'. При нажании на кнопку 'info' пользователь переходит на страницу для просмотра заметки (Рис. 7).



Рис. 6 : начальная страница пользователя.

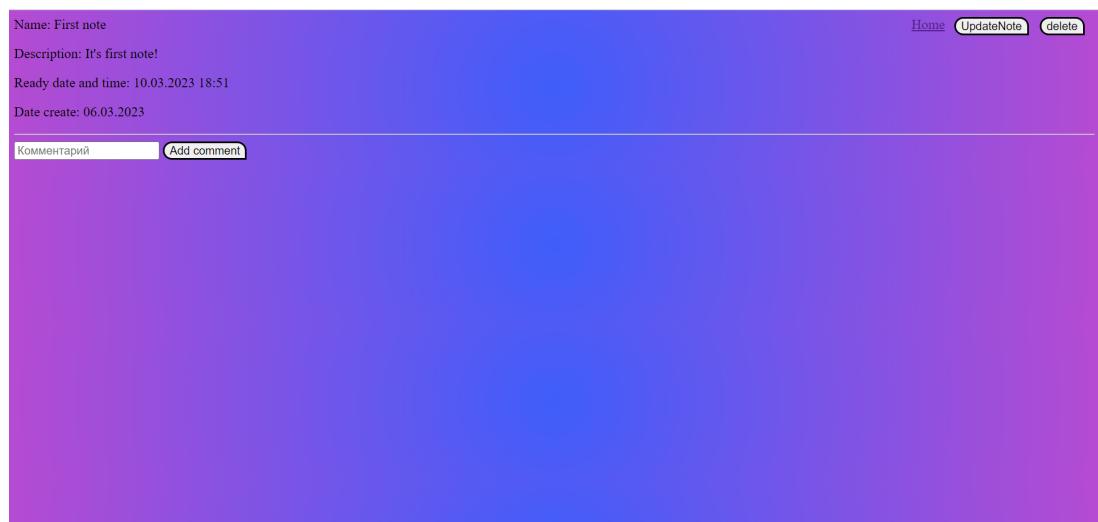


Рис. 7 : информация о заметке.

Пользователь может редактировать заметку, для этого нужно нажать на кнопку 'UpdateNote'. При нажатии на кнопку 'UpdateNote' пользователь переходит на страницу для редактирования заметки (Рис. 8).

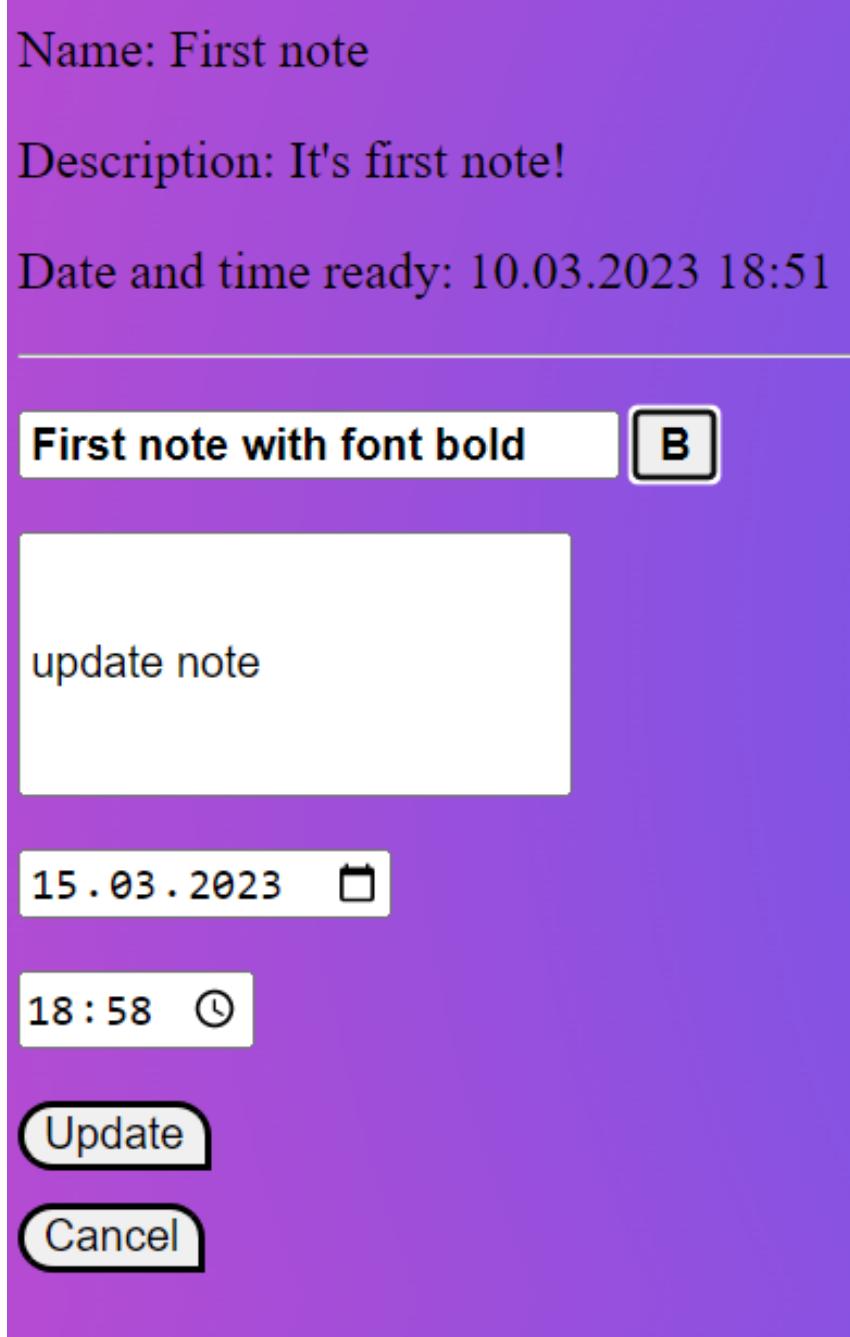


Рис. 8 : редактирование заметки.

При нажатии на кнопку 'Update' , пользователь автоматически переходит на страницу 'Информация о заметки' с обновленными данными.

Также пользователь может добавлять комментарии к заметке (Рис. 9).

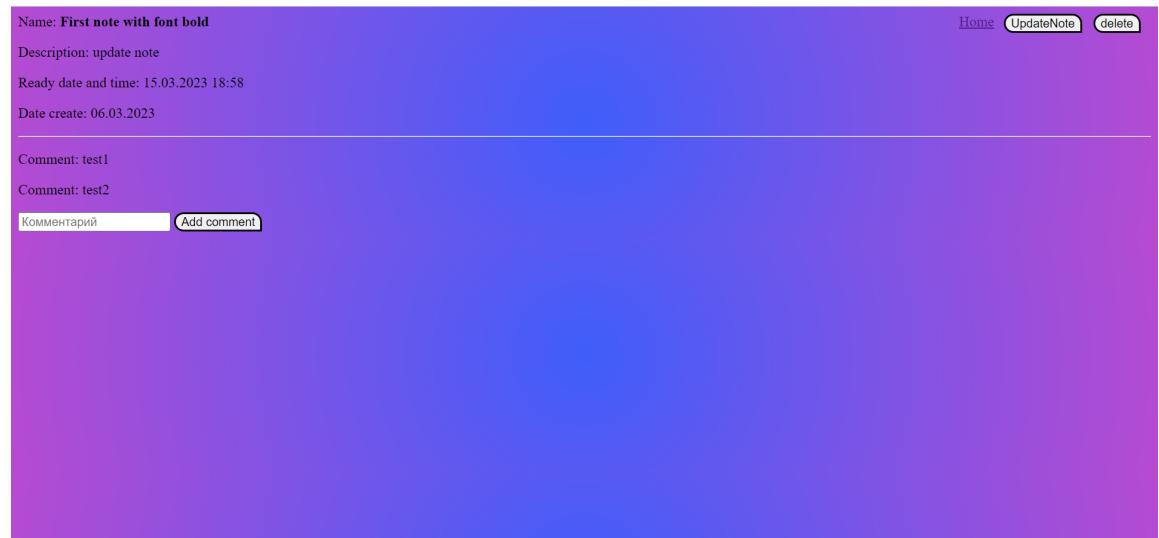


Рис. 9 : добавление комментария.

При нажатии на ссылку 'Home' пользователь переходит на начальную страницу.

Также есть всплывающее меню на начальной странице (Рис. 10), в которой можно сделать поиск по названию заметки, для этого нужно написать название и нажать на кнопку 'find'. После нажатия на кнопку 'find' пользователь перейдет на страницу найденных заметок (Рис. 11).

При нажатии на кнопку 'delete' на странице 'info' пользователь удаляет заметку, он его сможет увидеть только в корзине. Для того чтобы перейти в корзину, нужно нажать кнопку "trash". При нажатии на кнопку 'trash' пользователь сможет увидеть все удаленные заметки (Рис. 12).

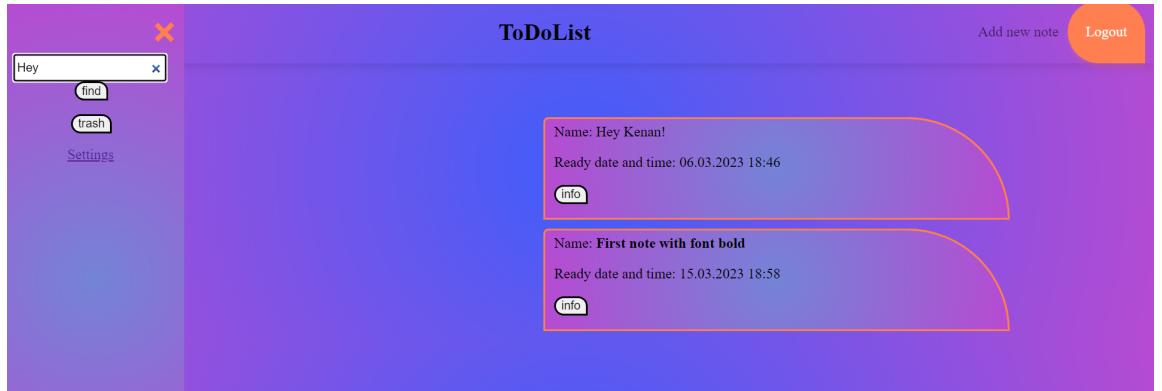


Рис. 10 : начальная страница.

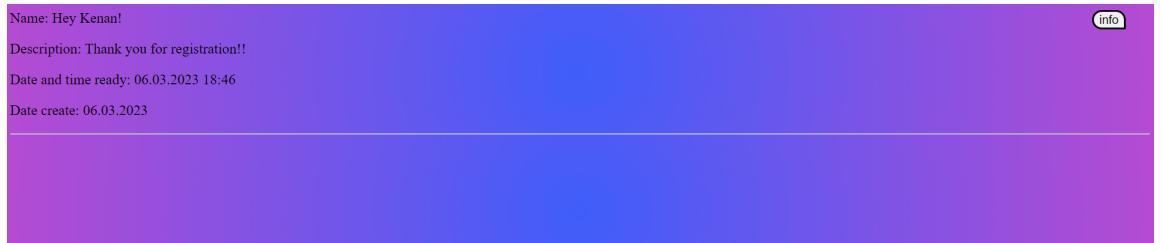


Рис. 11 : список найденных заметок.

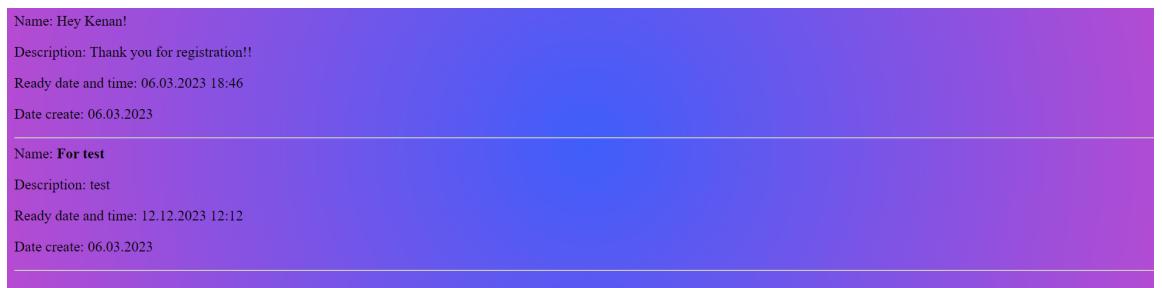


Рис. 12 : корзина.

Для просмотра информации о личных данных пользователь нажимает на ссылку 'Settings' во всплывающем окне (Рис. 13).

Для выхода из сервера пользователь нажимает на кнопку 'Logout' на начальной странице.

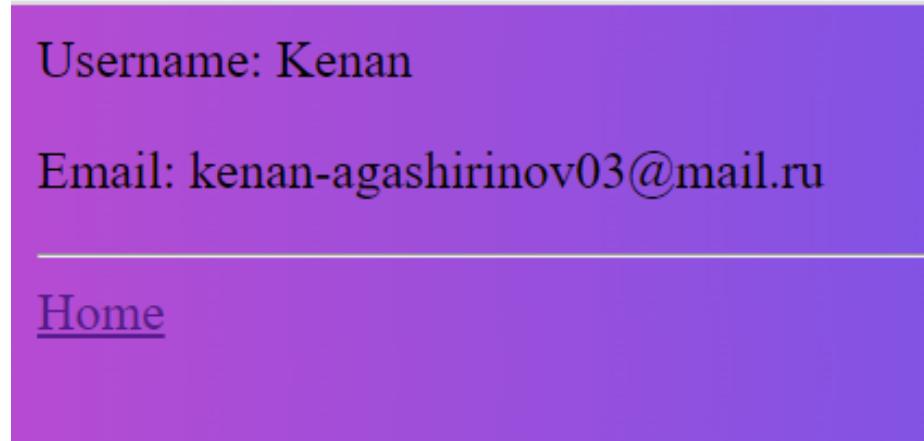


Рис. 13 : настройки.

#### 4. Архитектура программы

Графическая схема архитектуры программы показана на (Рис. 14). Архитектура данного приложения основана на принципе MVC (Model-View-Controller). View посыпает POST и GET запросы, далее их принимает Controller (1). Controller обеспечивает взаимодействие между пользователем и системой, контролирует и направляет данные от пользователя к системе и наоборот. После получение запросов из View, Controller создает определенную логику в Model (2). То есть, в Model мы обрабатываем запросы клиента. Как правило, данные в современных приложениях хранятся в базах данных. За получение данных из БД, их обработку и передачу этих самых данных обратно в Model, а затем во View отвечают Service(3) и Repository(4). В Repository мы напрямую обращаемся к БД (5) и получаем из нее необходимые данные (делаем запросы). А затем эти полученные данные направляются в Service (6), где они определенным образом обрабатываются и создается основная бизнес-логика проекта.

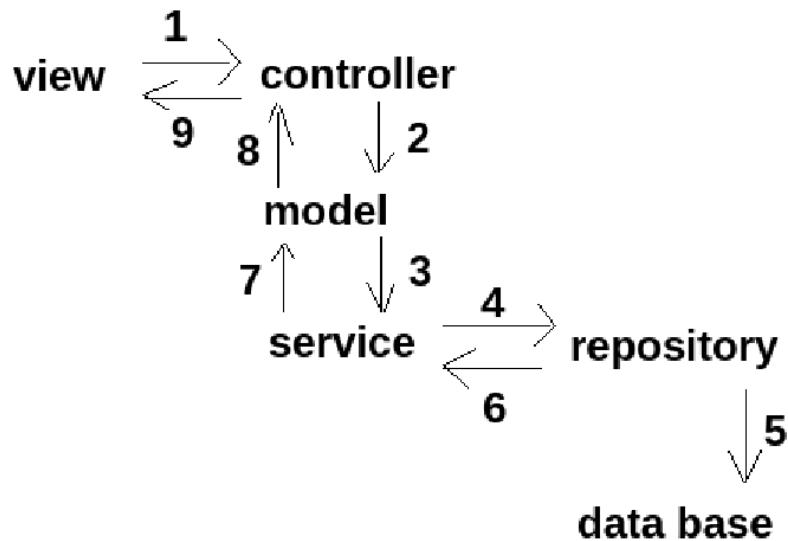


Рис. 14 : архитектура программы

## 5. Описание наиболее важных классов

### 1. Пакет config.

*Классы:*

- SecurityConfig - отвечает за конфигурирование авторизации, аутентификации и безопасности приложения.

### 2. Пакет controllers.

*Классы:*

- AuthController - в данном классе мы обрабатываем следующие запросы:
  - URL: '/auth/singIn' - возвращает страницу с формой для авторизации (GET-запрос).
  - URL: '/auth/login' - возвращает страницу с формой для регистрации пользователя (GET-запрос).
  - URL: '/auth/Login' - добавляет зарегистрировавшегося пользователя (POST-запрос).

- GET-запрос URL: '/auth/home' - возвращает начальную страницу со всеми мероприятиями (GET-запрос).
- MainController - в данном классе мы обрабатываем следующие запросы:
  - URL: '/home' - возвращает главную страницу с авторизацией и регистрацией (GET-запрос).
  - URL: '/newNote' - возвращает страницу с формой для заполнения новой заметки (GET-запрос).
  - URL: '/info' - возвращает подробную информацию о заметке (POST-запрос).
  - URL: '/updateNote' - возвращает информацию о заметки и форму для редактирования (GET-запрос).
  - URL: '/updateNote' - редактирует заметку и возвращает подробную информацию о заметке (POST-запрос).
  - URL: '/delete' - удаляет заметку (POST-запрос).
  - URL: '/trash' - возвращает все удаленные заметки (POST-запрос).
  - URL: '/comment' - добавляет комментарии к заметке (POST-запрос).
  - URL: '/findNote' - возвращает все найденные заметки (POST-запрос).
  - URL: '/settings' - возвращает данные о пользователе (GET-запрос).

### 3. Пакет models

Все классы, лежащие в данном пакете являются сущностями, то есть каждому классу соответствует одна таблица в БД.

*Классы:*

- Person - данный класс характеризует пользователя, имеет следующие поля: id, username, email, password.
- Note - данный класс характеризует заметки в колонке, имеет следующие поля: id, personId, notesName, description, timeAnDdate, createDate, time, bold.
- Comment - данный класс характеризует комментарии, имеет следующие поля: id, notesId, comments.
- Trash - данный класс характеризует корзину, имеет следующие поля: id, personId, notesName, description, timeAnDdate, createDate, time, bold.

#### 4. Пакет repositories

Все интерфейсы в данном пакете наследуются от интерфейса JpaRepository, который уже реализует за нас основные методы CRUD приложения (get, update, save, delete).

*Интерфейсы:*

- CommentRepository - данный интерфейс делает запросы к сущности Comment в БД.  
Основной метод:  
\* Comment findAllByNoteOwner() - находит все комментарии по заметкам.
- NoteRepository - данный интерфейс делает запросы к сущности Note в БД.  
Основной метод:  
\* List<Note> findById(int id); - используется для поиска заметки по id.
- PersonRepository - данный интерфейс делает запросы к сущности Person в БД.  
Основные методы:

- \* Optional<Person> findByUsername(String username);
  - возвращает список пользователей по username.
- \* Optional<Person> personRepository.findAll() - возвращает всех пользователей.
- TrashRepository - данный интерфейс делает запросы к сущности Trash в БД. Основной метод:
  - \* List<Trash> trashRepository.findAll(); - возвращает список удаленных заметок.