НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р. Е. Алексеева

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА к курсовой работе

*Спорышева Егора Максимовича*

*(фамилия, имя, отчество)*

Институт Институт радиоэлектроники и информационных технологий

Кафедра Графические информационные системы

Группа 21-ИСТ-4

День защиты « 27 » декабря 2022 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Р.Е. Алексеева

Кафедра Графические информационные системы

Заведующий кафедрой ГИС

(подпись) Филинских А.Д. *(фамилия и. о.)*

*(дата)*

# Разработка сайта на Angular

*«Галерея»*

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

**КОНСУЛЬТАНТЫ:**

1.По

*(подпись)* *(фамилия и. о.)*

*(дата)*

2.По

*(подпись)* *(фамилия и. о.)*

*(дата)*

3.По

*(подпись)* *(фамилия и. о.)*

*(дата)*

**РЕЦЕНЗЕНТ**

*(подпись)* *(фамилия и. о.)*

*дата)*

**РУКОВОДИТЕЛЬ:**

Соловьёв С.А.

*(подпись)* *(фамилия и. о.)*

*(дата)*

**СТУДЕНТ**

Спорышев Е. М.

*(подпись)* *(фамилия и. о.)*

21-ИСТ-4

*(дата)* *(группа или шифр)*

Проект защищен

Протокол №

С оценкой

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | КР – НГТУ – 090302 – 21-ИСТ-4 – 14 – ПЗ | | |
| Изм. | Лист | № докум. | Подпись | Дата | | | |
| Провер. | | Соловьёв С.А | | | ПОЯСНИТЕЛЬНАЯ ЗАПИСКА | Лит. | Лист 2 / Листов 39 |
| Разраб. | | Спорышев Е.М | | | | | Кафедра КТТП гр. 21-ИСТ-4 |

Нижегородский государственный технический университет им. Р.Е. Алексеева

Кафедра: Графические информационные системы

УТВЕРЖДАЮ
*Зав. кафедрой* **Филинских А.Д.**

З А Д А Н И Е
на курсовое проектирование

Студент: Спорышев Егор Максимович, группа 21-ИСТ-4

Тема курсового проекта: Разработка Web-приложения «Галерея»

Исходные данные к проекту: основные знания и сведения о Typescript как основном языке написания работы, сведения об Angular, Node.js, Webpack и json-server

Содержание графического материала:

чертежи:
1. Иллюстрации с примером оформления страниц сайта
2. Картинки из интернета

Содержание пояснительной записки:

Перечень вопросов, подлежащих разработке

Анализ исходных данных, и разработка ТЗ

Руководство пользователя

Руководство программиста

Листинг

Основная рекомендуемая литература:

[1] Руководство по Angular 14. Режим доступа https://metanit.com/web/angular2/

[2] Angular. Режим доступа https://angular.io/docs

| | | | | |
|---|---|---|---|---|
| | | | Руководитель | Соловьёв С.А. |

« 27 » декабря 2022 г.

Студент ФИО Спорышев Егор Максимович

Дата 27.12.2022

## Содержание

## Введение

Интернет с каждым годом развивается, создавая все новые и новые вещи, которые хотелось бы запечатлеть, оставить в памяти. Именно поэтому приложения, способные сохранять это, будут иметь популярность.

На данный момент в интернете множество сервисом по картинкам, но нет такого простого и понятного, в котором можно было бы с легкостью загружать картинки, распределять их по категориям, добавлять им описание, просматривать картинки других пользователей. В ходе данной работы необходимо разработать web- приложение, в котором будет реализован перечисленный функционал.

# 1. Анализ исходных данных, и разработка ТЗ

## 1.1. Назначение разработки

Разработать приложение «Галерея», которое предназначено для пользователей, желающих сохранять, просматривать картинки и распределять их по категориям.

## 1.2. Минимальные требования к составу и параметрам технических средств: ЭВМ, внешние устройства

Функционирование приложения обеспечивается следующими параметрами технических средств:

- операционная система Windows (начиная с 7 версии);
- процессор Intel Core i5;
- оперативная память не менее 4 гигабайт
- монитор
- устройство для ввода информации
- доступ в сеть Интернет.
- Возможность просмотра сайта должна осуществляться в любом современном браузере с поддержкой HTML5. Сайт должен быть работоспособен (информация, расположенная на нем, должна быть доступна).

## 1.3. Требования к информационной и программной совместимости

Для успешного дальнейшего выполнения курсовой работы требовалось установить необходимое ПО:

- Node.js — это кроссплатформенная среда с открытым исходным кодом для разработки серверных и сетевых приложений;
- Webpack — это статический модульный сборщик для приложений на JavaScript;
- Angular и Angular CLI — это фреймворк от компании Google для создания клиентских приложений, который нацелен на разработку одностраничных приложений (1);
- TypeScript — это расширенная версия языка JavaScript, главной целью которого является упрощение разработки крупных JS-приложений;
- VS Code — мощная IDE для веб-разработки;
- json сервер (2).
- .

## 1.4. Требования к функциональным характеристикам

В системе «Галерея» будут реализованы следующие возможности:

a) Регистрация и авторизация в системе
b) Добавление картинок
c) Отображение своих картинок и картинок других пользователей
d) Фильтрация картинок по категориям: «человек», «природа», «животные», «автомобили», «интерьер»
e) Отображения личных данных пользователя в профиле
f) Редактирование пользователя
g) Запись всех данных о пользователях, картинках.
h) Удаление аккаунта из базы данных

## 1.5. Выбор и обоснование языков программирования и используемых инструментальных средств

Для реализации данного приложения мною был задействован язык программирования TypeScript, который имеет ряд преимуществ над JavaScript – например, статическая типизация, лучшая поддержка в IDE, а также возможность разработки приложения с помощью новейших инструментов, не волнуясь о поддержке их браузером. В качестве среды разработки – Visual Studio Code. Выбор сделан с учетом удобного интерфейса, меньшей ресурсозатрачиваемости чем у других сред на рынке.

Для визуального представления был скачан и подключен в проект Bootstrap – фреймворк для разработки с помощью HTML, CSS и JS.

## 2. Внешняя спецификация

### 2.1. Разработка сематического ядра

Разработка семантического ядра это один из самых важных этапов в создании приложения, ведь именно по нему и составляется общая структура и структура отдельных страниц проекта.

Таблица 1. Семантическое ядро

| Ключевые слова | Частота | Фразовая частота | Точная частота |
|---|---|---|---|
| Картинка | 35 898 706 | 58120 | 57 919 |
| Галерея | 1 635 304 | 24010 | 23 945 |
| Картинка животное | 165 539 | 88 | 22 |
| Картинка человек | 284 955 | 314 | 314 |
| Картинка природа | 158 603 | 1400 | 1 310 |

На основе полученного семантического ядра мною было разработано представление будущей системы.

### 2.2. Разработка структуры приложения (карта)

Создаваемое реактивное приложение будет состоять из разделов:

1. Вход
2. Регистрация
3. О нас
4. Галерея
5. Мои картинки
6. Добавить картинку
7. Личный кабинет

Со страницы входа будет возможность перейти на страницу регистрации или, при успешной авторизации, на главную страницу. Внутри системы возможен переход между всеми страницами, с каждой страницы можно осуществить выход из приложения, при этом пользователь будет перенаправлен в окно авторизации.

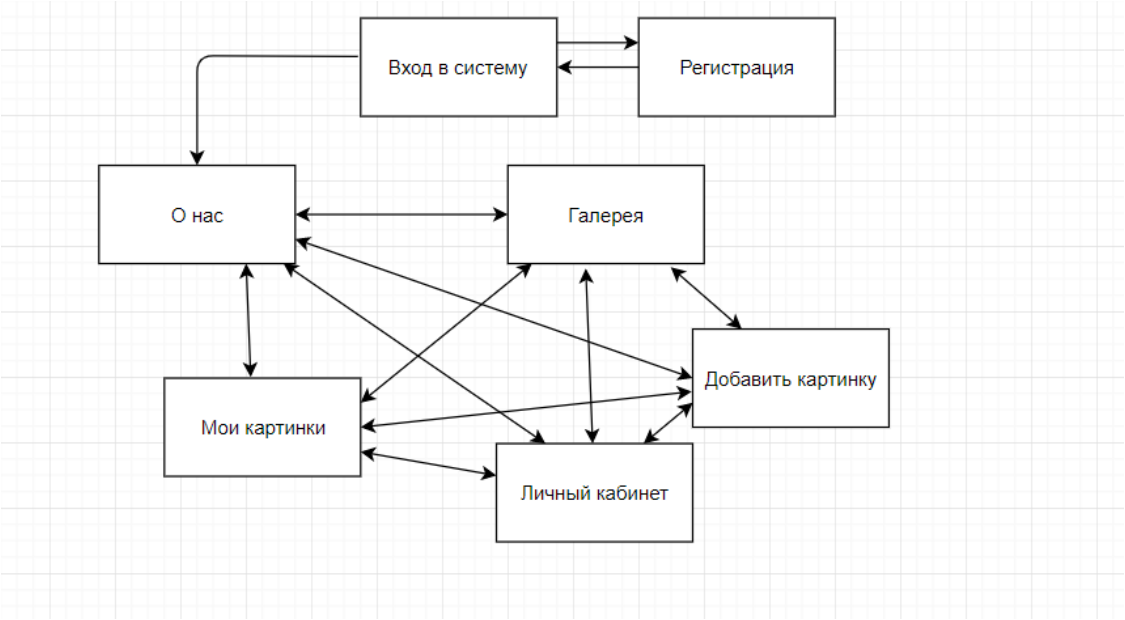


Рисунок 1 – Карта сайта

## 2.3. Функциональная схема

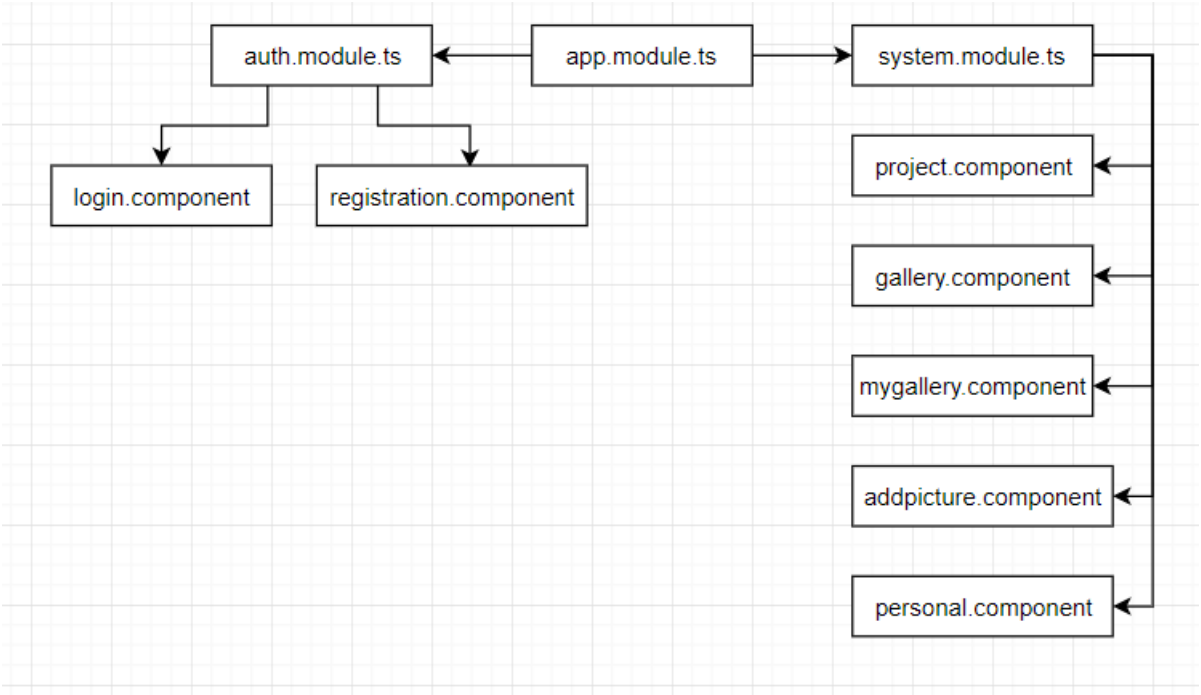


Рисунок 2 – Описание модулей приложения

# 3. Руководство пользователя

## 3.1. Назначение программы

Разработанное приложение предназначено для сохранения картинок по категориям. Предоставляет пользователю возможность добавлять картинку с категорией и описанием, просматривать свои картинки и картинки других пользователей.

## 3.2. Описание интерфейса

Требования к интерфейсу:

- интерфейс пользователя должен быть удобным, понятным и простым в использовании.
- должен обеспечивать с высокой скорости работы пользователя;
- быстрое обучение пользователя;
- обеспечение защиты от человеческих ошибок;
- субъективное удовлетворение пользователя.

После загрузки приложения открывается страница авторизации, с помощью которой пользователь может войти в систему или перейти на страницу регистрации.

После авторизации открывается страница «О нас», которая содержит три основных компонента: header (статичный), основную часть (изменяемая) и footer (статичный). На ней рассказывается о целях данного проекта.

Страница «Галерея» содержит все картинки пользователей, можно отображать их по категориям «Все» «человек», «природа», «животные», «автомобили», «интерьер». К каждой картинке представлено описание, создаваемое пользователем.

Страница «Мои картинки» содержит картинки конкретных пользователей с описанием, отображаемых по категориям.

На странице «Добавить картинку» можно добавить картинку, загрузив ее на сервер, указав категорию и описание.

На странице «Личный кабинет» отображаются данные авторизованного пользователя.

Директива, реализующая выпадающий список, состоит из двух пунктов: изменить профиль и выйти. На странице «Edit» возможно как изменить логин и пароль пользователя, так и удалить аккаунт.

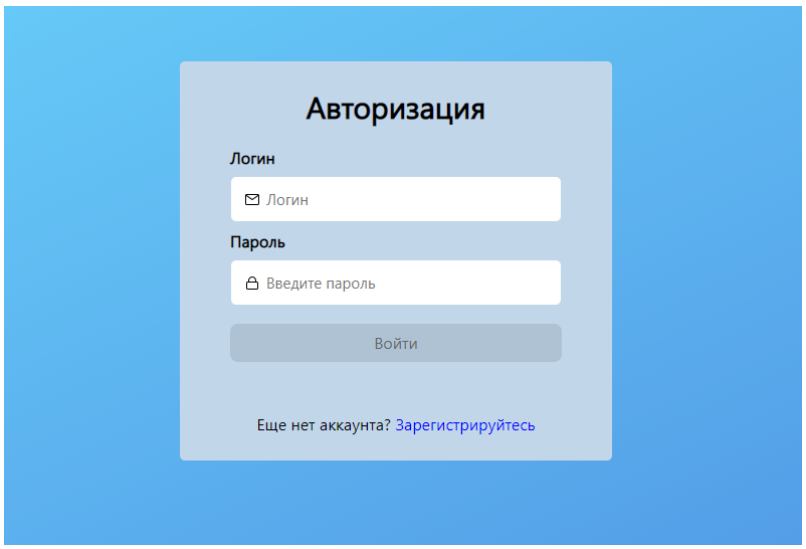


Рисунок 3 – форма авторизации

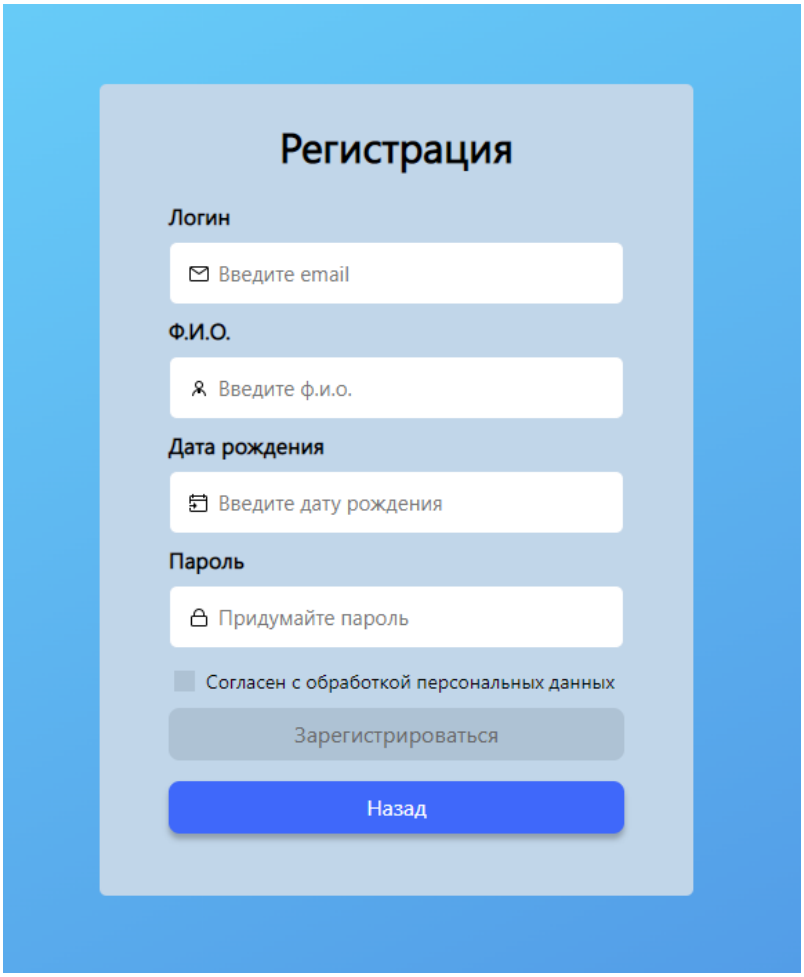


Рисунок 4 – форма регистрации

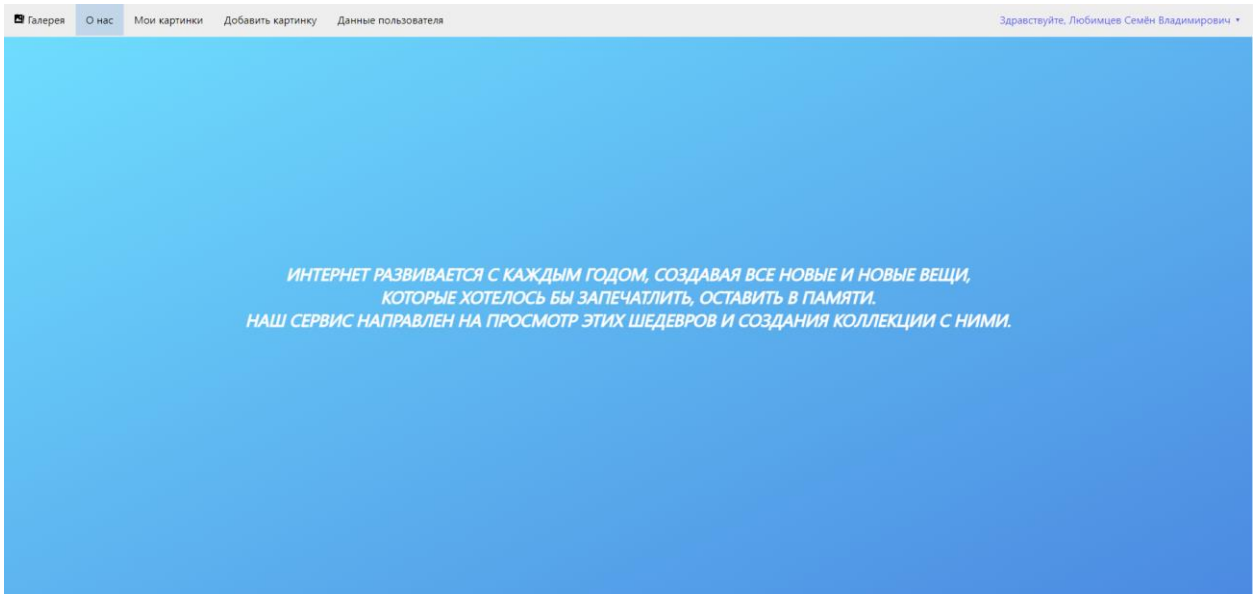


Рисунок 5 – страница «О нас»

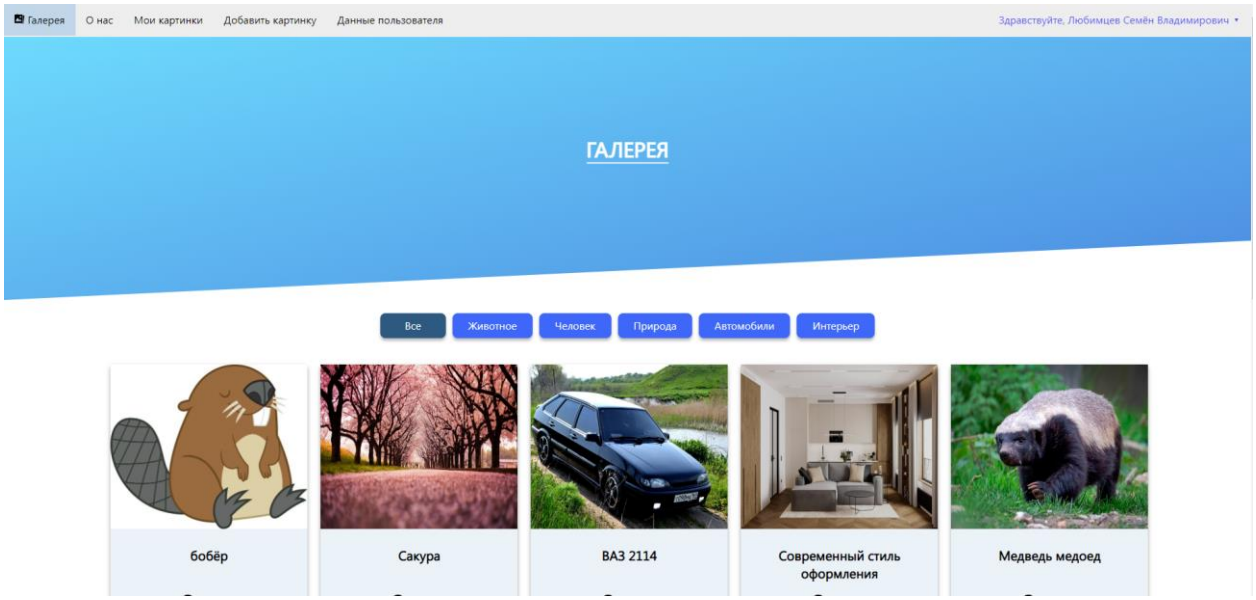


Рисунок 6 – страница «Галерея»

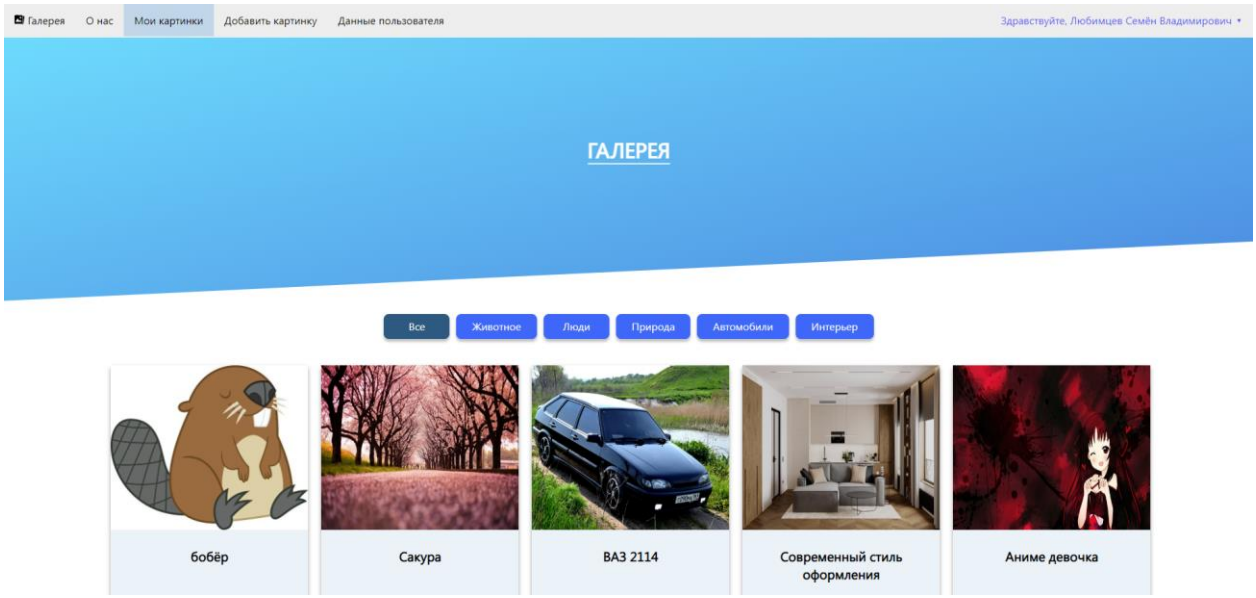


Рисунок 7 – страница «Мои картинки»

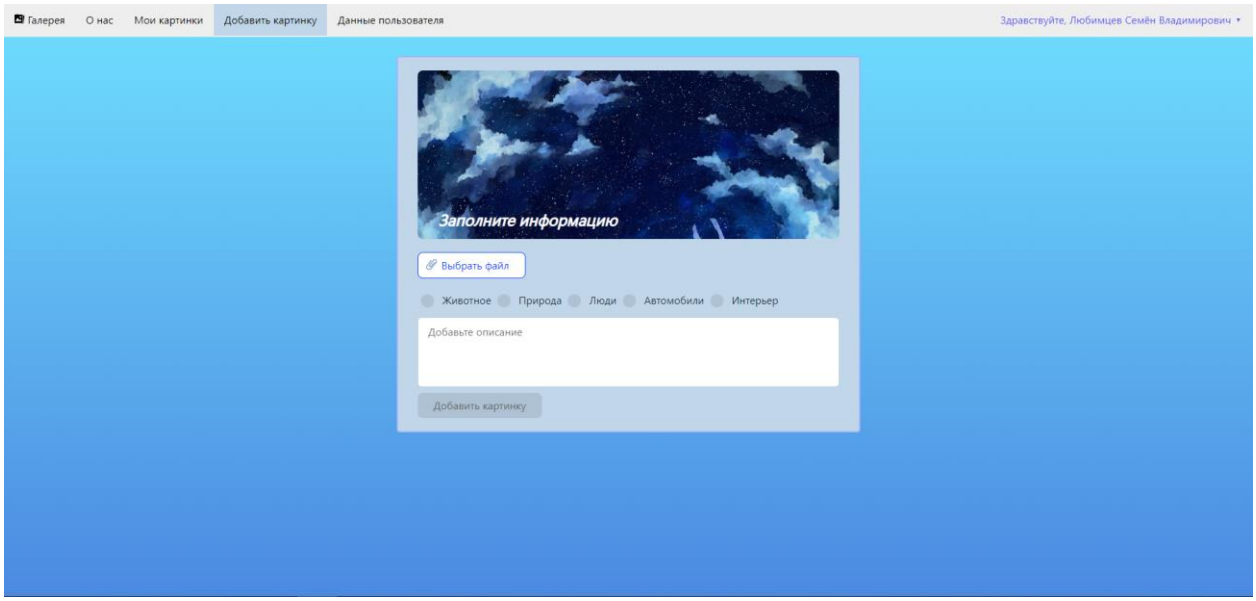


Рисунок 8 – страница «Добавить картинку»

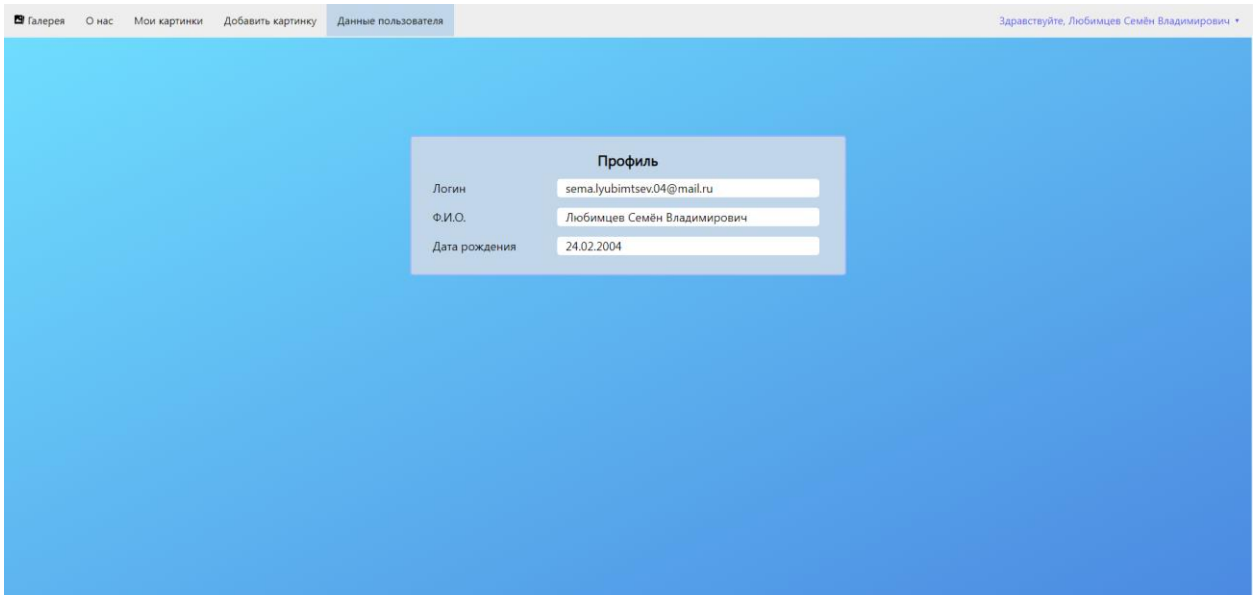


Рисунок 9 – страница «Личный кабинет»

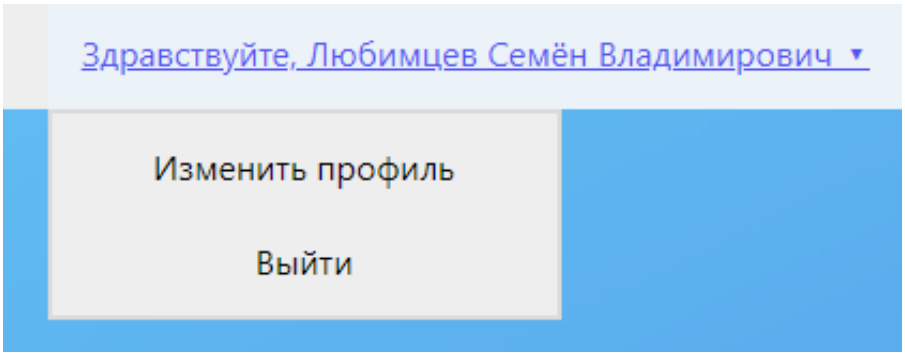


Рисунок 10 – страница «Директива»

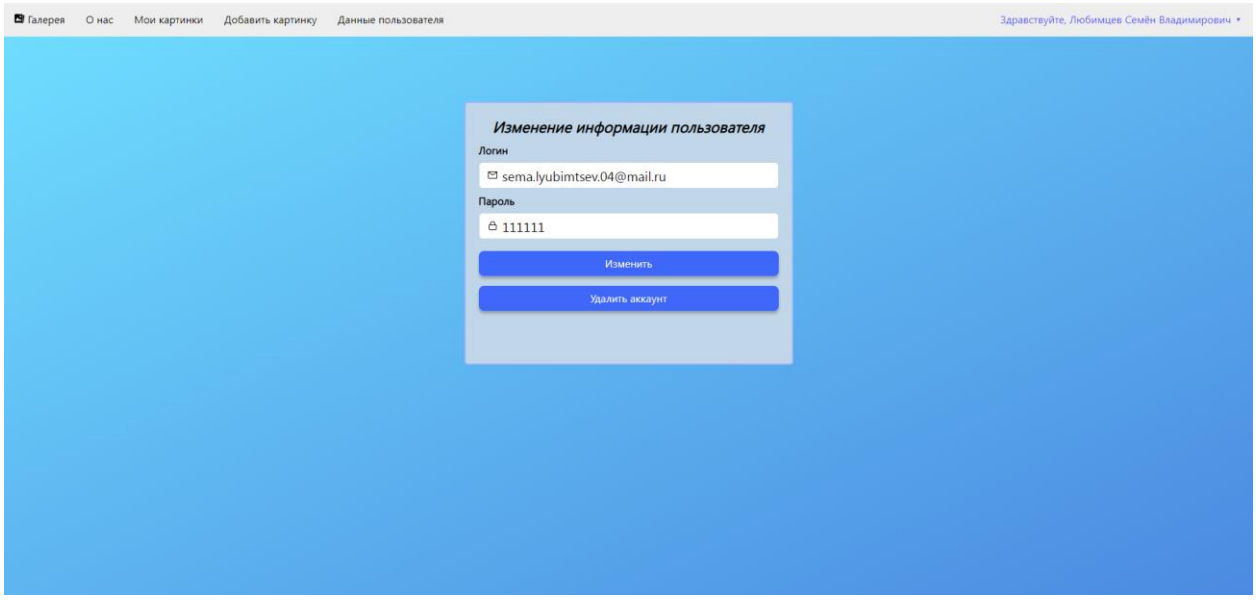


Рисунок 11 – страница «Изменить профиль»

### 3.3. Требования к входным данным

При авторизации поле Email не должно быть пустым и должно содержать корректный формат ввода почты. Пароль не должен быть пустым и должен иметь шесть и более символов.

При регистрации поле Email не должно быть пустым, должно содержать корректный формат ввода почты и не должно содержать ранее введенный другим пользователем email. Пароль не должен быть пустым и должен содержать шесть и более символов. Поле фио не должно быть пустым. Дата рождения не должна быть пустой, длина ее должна быть восемь и более символов.

# 4. Руководство программиста

## 4.1. Организация ввода данных в программу и вывода

Основными элементами на страницах нашей системы являются формы. Реактивные формы (Angular reactive forms) построены на основе механизма, использующего реактивный подход к программированию. Для их использования нужно импортировать модуль ReactiveFormsModule. Создание и валидация форм осуществляется прямо в компоненте, что делает работу с данными более гибкой (3).

Информация, используемая в системе, хранится в db.json – файле БД, в котором хранится массив пользователей и массив постов.

Вывод данных осуществляется с помощью двусторонней связки, директивы ngFor и get запроса.

```
 <div class="backgr">
  <form
  [formGroup]="form"
  (ngSubmit)="onSubmit()"
  >

  <div class="text">Введите свои данные</div>
  <div class="container">
    <div
      class="form-group"
      [ngClass]="{'has-error': form.get('email')?.invalid &&
form.get('email')?.touched}"
    >
      <div class="input-group">

        <label for="psw"><div class="formtext">Логин  <span
><i></i></span></div></label>
        <input
        type="text"
         placeholder="Логин"
         formControlName="email"
         >
         <img class="icon" src="https://img.icons8.com/small/16/000000/secured-
letter.png"/>
      </div>
      <div
        class="alert alert-{{massage.type}}"
        *ngIf="massage.text">
```

```
        {{massage.text}}
      </div>
      <span
        class="form-help-text"
        *ngIf="form.get('email')?.invalid && form.get('email')?.touched"
      >
            <span *ngIf="form.get('email')?.errors?.required"> Email не может
быть пустым</span>
            <span *ngIf="form.get('email')?.errors?.email"> Email не соответ-
ствует стандарту</span>
          </span>
    </div>
    <div >
      <div class="input-group">

        <label for="psw"><div class="formtext">Пароль  <span
><i></i></span></div></label>
        <input
          type="password"

          placeholder="Введите пароль"
          formControlName="password"
        >
        <img class="icon" src="https://img.icons8.com/small/16/000000/lock.png"/>
      </div>
      <span
        class="form-help-text"
        *ngIf="form.get('password')?.invalid && form.get('password')?.touched"
      >
            <span *ngIf="form.get('password')?.errors?.required"> Пароль не может
быть пустым</span>
            <span *ngIf="form.get('password')?.errors?.minlength &&
form.get('password')?.errors?.minlength?.requiredLength">
        Пароль должен быть больше {{form.get('password')?.errors?.minlength?.re-
quiredLength}}
              символов. Сейчас {{form.get('password')?.errors?.minlength?.actual-
Length}}.
      </span>
          </span>
      </div>
      <app-button
        text="Войти"
        type="submit"
        [disabled]="form.invalid"
        [buttonwhite]="true"
        [fullwidth]="true"
      >
      </app-button>
      <p>Еще нет аккаунта?<a routerLink="/registration" routerLinkActive="ac-
tive"> Зарегистрируйтесь</a></p>
```

```
    </div>
    </form>
</div>
```

Рисунок 18 – ввод данных в форме авторизации

```
<div class="space">
  <div class="tabl">
    <div class="card-image">
      <h2 class="card-heading">
        Заполните информацию
      </h2>
    </div>
    <img class="icon" src="https://img.icons8.com/officexs/16/000000/at-
tach.png" />
    <div class="buttons">
      <app-button text="Выбрать файл" (click)="onOpenFileDialog()" [button-
white]="true" [buttonadd]="true"></app-button>
    </div>
    <input id="file-input" style="display: none" type="file" (change)="file-
Change($event)" placeholder="Upload file" />
    <form #f="ngForm" (ngSubmit)="onSubmit(f)">
      <div class="alert alert-{{massage.type}}" *ngIf="massage.text">
        {{massage.text}}
      </div>
      <div class="form-group" [ngClass]="{'has-error': category?.invalid &&
category?.touched}">

        <p>
          <input type="radio" value="animal" name="category" ngModel #cate-
gory="ngModel" required
                 class="option-input radio"> Животное
          <input type="radio" value="nature" name="category" ngModel #cate-
gory="ngModel" required
                 class="option-input radio"> Природа
          <input type="radio" value="human" name="category" ngModel #cate-
gory="ngModel" required
                 class="option-input radio"> Люди
          <input type="radio" value="car" name="category" ngModel #cate-
gory="ngModel" required
                 class="option-input radio"> Автомобили
          <input type="radio" value="interior" name="category" ngModel #cate-
gory="ngModel" required
                 class="option-input radio"> Интерьер
        </p>
        <span *ngIf="category?.invalid && category?.touched">
          <span>Поле обязательно для заполнения</span>
        </span>
      </div>
      <div class="form-group" [ngClass]="{'has-error': caption?.invalid &&
caption?.touched}">
        <textarea cols="80" rows="4" placeholder="Добавьте описание" ngModel
```

```
name="caption" #caption="ngModel"
          required></textarea>
        <span *ngIf="caption?.invalid && caption?.touched">
          <span class="form-help-text">Поле обязательно для заполнения</span>
        </span>
      </div>
      <div class="buttons">
        <app-button [buttonwhite]="true" text="Добавить картинку" type="sub-
mit" [disabled]="f.invalid"
          style="font-size: 20px"></app-button>
      </div>
    </form>
  </div>
</div>
```

Рисунок 19 – ввод данных в форму для добавления картинки

```
<div class="space">


  <div class="alert alert-{{massage.type}}" *ngIf="massage.text">
    {{massage.text}}
  </div>

  <form #f="ngForm" (ngSubmit)="onSubmit(f)">
    <div class="centerText">Изменение информации пользователя</div>

      <div class="input-group" [ngClass]="{'has-error': email?.invalid &&
email?.touched}">
        <label for="uname">
          <div class="formtext">Логин <span class="input-
group"><i></i></span></div>
        </label>
        <input type="text" class="form-control" placeholder="Введите новый email"
ngModel name="email" #email="ngModel"
          required [ngModel]="currentUser.email">
          <img class="icon" src="https://img.icons8.com/small/16/000000/secured-
letter.png"/>
      </div>
      <span *ngIf="email?.invalid && email?.touched">
        <span>
          Поле обязательно для заполнения
        </span>
      </span>

    <div class="form-group" [ngClass]="{'has-error': password?.invalid && pass-
word?.touched}">
      <div class="input-group">
        <label for="psw">
          <div class="formtext">Пароль <span class="input-
group"><i></i></span></div>
        </label>
```

```
        <input type="text" class="form-control" placeholder="Придумайте новый па-
роль" ngModel name="password"
          #password="ngModel" required [ngModel]="currentUser.password">
          <img class="icon"
src="https://img.icons8.com/small/16/000000/lock.png"/>
      </div>
      <span class="form-help-text" *ngIf="password?.invalid && pass-
word?.touched">
        <span>
          Поле обязательно для заполнения
        </span>
      </span>
    </div>
    <div class="buttons">
      <app-button type="submit" [disabled]="f.invalid" text="Изменить" [but-
tonblue]="true" [fullwidth]="true">
      </app-button>

      <app-button [buttonblue]="true" [fullwidth]="true" type="submit"
text="Удалить аккаунт" (click)="delete()">
      </app-button>
    </div>
  </form>
</div>
```

Рисунок 20 – вывод данных для изменения профиля

## 4.2. Описание шаблона web-приложения (HTML5/ SCSS)

Шаблоны можно разделить на две группы: шаблоны регистрации и авторизации, шаблон для добавления картинки, шаблон ввода новых данных в профиле.

В самом главном шаблоне приложения app.component.html прописываем <router-outlet></router-outlet>.

В шаблонах авторизации и регистрации прописаны формы для авторизации и регистрации, связанные их с соответствующими компонентами, так же

делаем валидаторы, для корректного ввода имени, email и пароля, и блокировки формы, при некорректном вводе.

### 4.3. Описание компонентов и модулей системы (TypeScript)

В модуле приложения app.module.ts прописаны модули для взаимодействия с системой. В декораторе NgModule, в массиве declarations подключён AppComponent, в массиве imports – остальные модули В app-routing.module.ts создан массив роутов, экспортирован и импортирован RouterModule в декораторе.

Компоненты и модули можно разделить на компоненты и модули регистрации и авторизации, страниц о нас, галерея, мои картинки, добавить картинку, личного кабинета.

Директория Auth включается в себя две директории login и registr, которыми являются, соответственно, компонент Авторизации и компонент Регистрации, а также главный модуль auth.module.ts, в котором хранится компоненты регистрации и авторизации. Этот модуль подключается к главному модулю app.module.ts. В auth-routing.module.ts прописаны пути на страницу логина и регистрации.

В компонентах авторизации и регистрации прописана логика для работы со страницами авторизации и регистрации: вывод сообщения об успешной регистрации с помощью метода showMessage, при неудачной попытке логина, с помощью того же метода выдаёт сообщение об ошибке, если вы ввели неверный пароль или email. Получение данных пользователя из базы данных при авторизации и добавление нового пользователя в базу данных при регистрации с помощью метода onSubmit().

Компоненты и модули страниц работают по одному принципу, прописаны пути до стилей и .html файлов в декораторе, так же с любой страницы можно перейти на другие.

### 4.4. Структура программы

Проект состоит из главной директории app, в которой находится компонент декоратор, так же в модуле массив роутов, который перенаправляет на страницу истории.

Директива auth состоит из директив:

- login - располагаются шаблон, css-файл и компонент авторизации;
- registration - располагаются шаблон, css-файл и компонент регистрации;

Профиль передаёт всё в директиву System, из неё подключается остальное:

- project – описывается, о чем проект;
- galley– описываются картинки всех пользователей;
- mygallery– описывается картинки конкретного пользователя;
- addpicture - описывается добавление картинки;

Директива shared состоит:

- models - описываются модели: пользователя,
- pipes – описывается фильтрация;
- services, - описываются сервисы: авторизации, отзывов и изменения пользователя;
-components – описываются компоненты кнопок и поля ввода

Структура страницы размечена с помощью HTML, стили страницы прописаны в CSS-файле, логика страниц прописана с помощью TypeScript.

### 4.5. Описание взаимодействий между сущностями web-приложения

Взаимодействие между страницами осуществляется с помощью роутов.

В файле app-routing.module.ts прописан redirect на страницу авторизации. В файле auth-routing.module.ts прописаны пути к страницам логина и регистрации. В файле system-routing.module.ts прописаны пути к страницам системы.

```
const routes: Routes = [
  {path: '', component: AuthComponent, children: [
    {path: 'login', component: LoginComponent},
    {path: 'registration', component: RegistrationComponent}
  ]}
];
```

Рисунок 23 – Файл auth-routing.module.ts

```
const routes: Routes = [
  {path:'', redirectTo:'login', pathMatch:'full'}
];
```

Рисунок 24 – Файл app-routing.module.ts

```
const routes: Routes = [
  { path: 'system', component: SystemComponent, children: [
    {path:'project', component: ProjectComponent},
    {path:'gallery', component: GalleryComponent},
    {path:'mygallery',component:MygalleryComponent},
    {path:'addpicture',component:AddpictureComponent},
    {path:'personal',component:PersonalComponent},
    {path:'edit',component:EditComponent}
  ]},
  {path: 'image/:id', component: ImageComponent},
];
```

Рисунок 25 – Файл system-routing.module.ts

## 5. Тестовый пример

Протестируем страницы Логина и регистрации, с помощью которой, пользователь регистрируется и получает доступ к приложению. Первым пользователь увидит форму входа(логина) – рисунок 26, с соответствующими сообщениями, если он некорректно ввёл email или пароль. Если аккаунта не будет в базе данных, нужно зарегистрироваться, по предложению ниже (4). Форма регистрации выглядит, как на рисунке 27, с соответствующими сообщениями, если он некорректно ввёл email или пароль, или повторил его неверно. После успешной регистрации, нас перекидывает на страницу профиля и в базу данных записываются данные об аккаунте, как на рисунке 28.

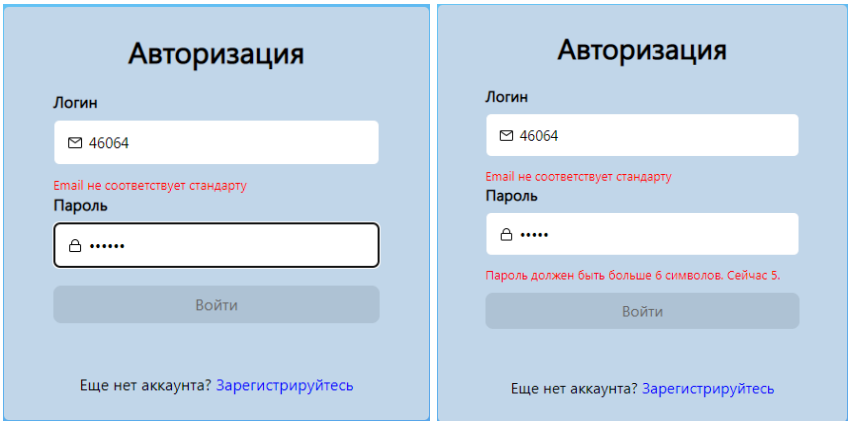


Рисунок 26 – Форма для авторизации

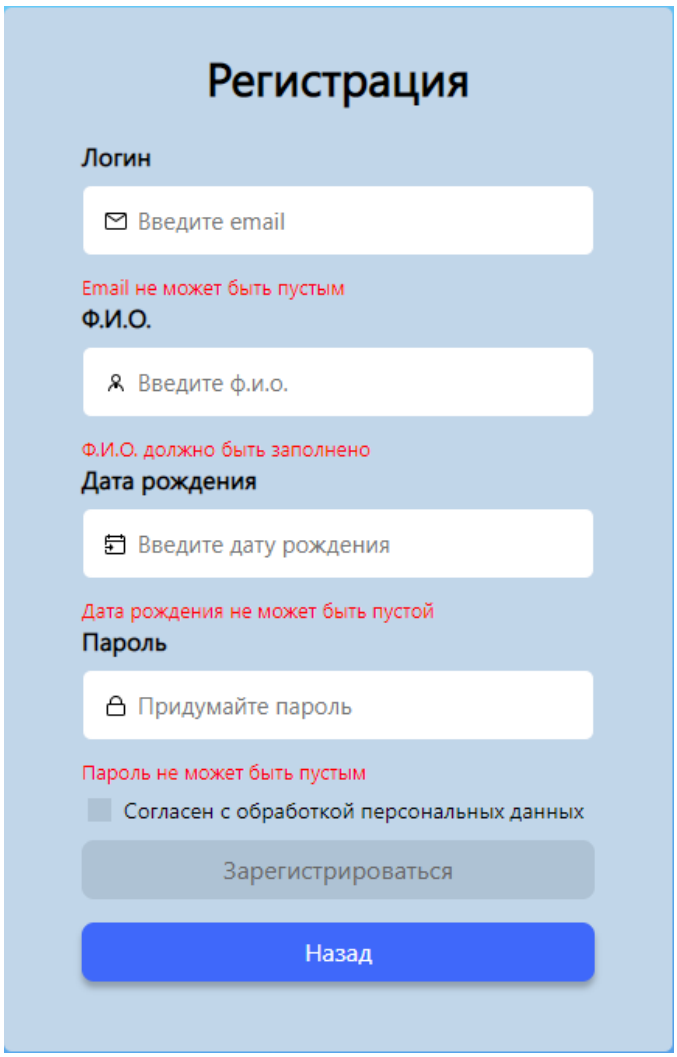


Рисунок 27 – Форма для регистрации


```
"users": [
  {
    "email": "sema.lyubimtsev.04@mail.ru",
    "password": "111111",
    "fio": "Любимцев Семён Владимирович",
    "datebirth": "24.02.2004",
    "id": 2
  },
```


Рисунок 28 – Данные в базе данных

## Заключение

В ходе курсовой работы было разработано приложение «Галерея». Программа позволяет зарегистрироваться в системе и войти в неё, предоставляет информацию о картинках, а также позволяет добавлять картинки и изменять данные в профиле.

К достоинствам системы можно отнести простоту работы с приложением, удобство в использовании, приятный интерфейс.

Планируется в дальнейшем доработка данного проекта: добавится возможность добавление своих собственных категорий. Кроме того, после добавления других страниц этот проект может развиться в более серьезное многопользовательское приложение, стать полноценно функционирующим сайтом.

В целом задание, поставленное на курсовую работу, выполнено в полном объеме. Проект полностью соответствует техническому заданию.

## СПИСОК ЛИТЕРАТУРЫ


1) *Руководство по Angular 14*. Получено 28.09.2020 г., из https://metanit.com/web/angular2/
2) *json-server*. (б.д.). Получено 03.11.2020 г., из GitHub json-server: https://github.com/typicode/json-server
3) (на языке оригинала), *Angular*. Получено 24.09.2020 г., из https://angular.io/
4) *Учебно-методическое пособие к выполнению лабораторных работ для студентов направления 09.03.02 - Информационные системы и технологии.* (2020). Нижний Новгород: Нижегородский государственный технический университет им. Р.Е. Алексеева (кафедра ГИС).

# ПРИЛОЖЕНИЕ

**AppModule.ts**

```
import { SharedModule } from './shared/shared.module';
import {UserService} from "./shared/services/user.service";
import { SystemComponent } from './system/system.component';
import { SystemModule } from './system/system.module';
import { AuthService } from './shared/services/auth.service';
import { ImageService } from './shared/services/image.service';
import { ImageFilterPipe } from './shared/services/filter.pipe';
import { ErrorComponent } from './error/error.component';

const appRoutes: Routes = [
  { path: '**', component: ErrorComponent }
];

@NgModule({
  declarations: [
    AppComponent,
    AuthComponent,
    SystemComponent,
    ErrorComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    AuthModule,
    HttpClientModule,
    SharedModule,
    SystemModule,
    RouterModule.forRoot(appRoutes)
  ],
  providers: [UserService,AuthService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**AuthModule.ts**

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

import { AuthRoutingModule } from './auth-routing.module';
import { LoginComponent } from './login/login.component';
import { RegistrationComponent } from './registration/registration.component';
import { SharedModule } from '../shared/shared.module';



@NgModule({
  declarations: [
    LoginComponent,
    RegistrationComponent,
```

```
  ],
  imports: [
    CommonModule,
    AuthRoutingModule,
    SharedModule
  ]
})
export class AuthModule { }
```

**login.component.ts**

```
import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { UserService } from '../../shared/services/user.service';
import { User } from '../../shared/model/user.model';
import { ActivatedRoute, Params, Router } from '@angular/router';
import { AuthService } from '../../shared/services/auth.service';
import { Massage } from '../../shared/model/massage.model';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss'],
})
export class LoginComponent implements OnInit {
  // @ts-ignore
  form: FormGroup;
  // @ts-ignore
  massage: Massage;

  constructor(
    private userService: UserService,
    private authService: AuthService,
    private router: Router,
    private route: ActivatedRoute
  ) {}

  ngOnInit(): void {
    this.massage = new Massage('danger', '');
    this.route.queryParams.subscribe((params: Params) => {
      if (params['canLogin']) {
        this.showMassage({
          text: 'Регистрация успешно завершена',
          type: 'success',
        });
      }
    });

    this.form = new FormGroup({
      email: new FormControl(null, [Validators.required, Validators.email]),
      password: new FormControl(null, [
        Validators.required,
        Validators.minLength(6),
```

```
      ]),
    });
  }

  private showMassage(massage: Massage) {
    this.massage = massage;

    window.setTimeout(() => {
      this.massage.text = '';
    }, 5000);
  }

  onSubmit() {
    console.log(this.form.get('email'));
    const formData = this.form.value;
    this.userService
      .getUsers(formData.email)
      .subscribe((user: User | undefined) => {
        if (user) {
          if (user.password === formData.password) {
            window.localStorage.setItem('user', JSON.stringify(user));
            this.authService.login();
            console.log('Вы успешно авторизованы');
            this.router.navigate(['system', 'project']);
          } else {
            this.showMassage({
              text: 'Пароль введён неверно',
              type: 'danger',
            });
          }
        } else {
          this.showMassage({
            text: 'Такого пользователя не существует',
            type: 'danger',
          });
        }
      });
  }
}
```

registration.component.ts

```
import { Component, OnInit } from '@angular/core';
import {FormControl, FormGroup, Validators} from "@angular/forms";
import {User} from '../../shared/model/user.model';
import {UserService} from '../../shared/services/user.service';
import {Router} from '@angular/router';
import { AbstractControl } from '@angular/forms';
@Component({
  selector: 'app-registration',
  templateUrl: './registration.component.html',
  styleUrls: ['./registration.component.scss']
})
```

```
export class RegistrationComponent implements OnInit {
  //@ts-ignore
  form: FormGroup;

  constructor(
    private userService: UserService,
    private router: Router
  ) { }

  ngOnInit(): void {
    this.form=new FormGroup({
      'email': new FormControl(null, [Validators.required, Validators.email], [this.forbid-
denEmails.bind(this)]),
      'fio':new FormControl(null, [Validators.required]),
      'password':new FormControl(null, [Validators.required, Validators.minLength(6)]),
      'agree': new FormControl(null, [Validators.requiredTrue]),
      'datebirth':new FormControl(null, [Validators.required, Validators.minLength(8)]),
    });

  }
  onSubmit() {
    const {email, password, fio, datebirth } = this.form.value;
    const user = new User(email, password, fio, datebirth);

    this.userService.createUser(user)
      .subscribe(() => {
        this.router.navigate(['/login'],{
          queryParams: {
            canLogin: true
          }
        });
      });
  }
  forbiddenEmails(control: AbstractControl): Promise<any>
  {
    return new Promise((resolve, reject )=> {
      this.userService.getUsers(control.value)
        .subscribe((user:User | undefined) => {
          if(user) {
            resolve({forbiddenEmail: true});
          }
          else {
            resolve(null);
          }
        });
    });
  }
  back():void
  {
    this.router.navigate(['/login']);
  }
```

```
}
```

**Addpicture.ts**

```
import { Component, OnInit, EventEmitter, Output } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Massage } from 'src/app/shared/model/massage.model';
import { NgForm } from '@angular/forms';
import { ImageService } from 'src/app/shared/services/image.service';
import { Imagee } from 'src/app/shared/model/image.model';
import { User } from 'src/app/shared/model/user.model';
import axios from 'axios';
import { UnloadService } from 'src/app/shared/services/unload.service';
@Component({
  selector: 'app-addpicture',
  templateUrl: './addpicture.component.html',
  styleUrls: ['./addpicture.component.scss']
})
export class AddpictureComponent implements OnInit {
  //@ts-ignore
  user:User;
  image:any;
  //@ts-ignore
  massage: Massage;
  @Output() onPictureAdd = new EventEmitter <Imagee>();

  constructor(private addpictureService: ImageService, private unloadService: UnloadService) {
}

  ngOnInit(): void {
    this.user = JSON.parse(window.localStorage.getItem('user') || '{}');
    this.massage = new Massage('success', '');
  }
  private showMassage(massage: Massage){
    this.massage = massage;
    window.setTimeout(()=>{
      this.massage.text = '';
    }, 5000);
  }
  onOpenFileDialog() {
    document.getElementById('file-input')?.click();
  }

  fileChange(event: any) {
    console.log(event);
    let fileList: FileList = event.target.files;
    if (!fileList.length) {
      return;
    }

    this.image= fileList[0];
```

```
  }

  async onSubmit(form: NgForm)
  {
    try{
      const { data } = await this.unloadService.unload(this.image);
      const {category, caption} = form.value;
      const picture = new Imagee(data,this.user.fio,category,caption);
      this.addpictureService.createImage(picture)
        .subscribe(()=>{
          this.showMassage({
            text:'Картинка успешно добавлена!',
            type: 'success'
          });
          form.reset();
        });
    }
    catch(error){
      console.log(error);
    }

  }

}
```

**Edit.component.ts**

```
import { Component, OnInit,Input, Output, EventEmitter } from '@angular/core';
import { User } from 'src/app/shared/model/user.model';
import { Massage } from 'src/app/shared/model/massage.model';
import { AuthService } from 'src/app/shared/services/auth.service';
import { NgForm } from '@angular/forms';
import { EditService } from 'src/app/shared/services/edit.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-edit',
  templateUrl: './edit.component.html',
  styleUrls: ['./edit.component.scss']
})
export class EditComponent implements OnInit {
  //@ts-ignore
  currentUser: User;
  //@ts-ignore
  massage:Massage;
  //@ts-ignore
  user: User;

  @Output() onRecordEdit = new EventEmitter<User>();

  constructor(private authServise: AuthService, private editService: EditService, private router:
Router) {
   }
```

```
  ngOnInit(): void {
    this.user = JSON.parse(window.localStorage.getItem('user') || '{}');
    this.currentUser  = this.user;
    this.massage = new Massage('success', '')
  }

  delete():void
  {
    this.editService.deleteUser(this.user.id).subscribe();
    this.router.navigate(['/login']);
  }

  onSubmit(form: NgForm) {
    const {email, password} = form.value;
    const newUser = new User(email, password, this.user.fio, this.user.datebirth, this.user.id);

    this.editService.updateUser(newUser)
      .subscribe((data: User) => {
        this.onRecordEdit.emit(data);
        this.massage.text = 'Аккаунт успешно обновлен!';
        window.setTimeout(() => this.massage.text = '', 5000)
      });
  }
}
```

**Gallery.component.ts**

```
import { Component, OnInit, Input} from '@angular/core';
import { ImageService } from 'src/app/shared/services/image.service';
import { Imagee } from 'src/app/shared/model/image.model';

@Component({
  selector: 'app-gallery',
  templateUrl: './gallery.component.html',
  styleUrls: ['./gallery.component.scss']
})
export class GalleryComponent implements OnInit {

  public visibleImages: Imagee[];

  // Create an input
  @Input() filterBy: string = 'all';

  constructor(private imageService: ImageService) {
    this.visibleImages=[];
    this.imageService.getImages().subscribe(res => {
      this.visibleImages=res;
    })

  }

  ngOnChanges() {
     //this.visibleImages = this.imageService.getImages();
```

```
    this.imageService.getImages().subscribe(res => {
      this.visibleImages=res;
    })
  }

  ngOnInit(): void {
    //this.imageService.getImages().subscribe(res=>
      //{this.visibleImages=res;})
  }
  concatUrl(url:string){
    return `http://localhost:3002/${url}`
  }
  currentFilter = { title: 'Все', value: 'all' };
  filters = [
    { title: 'Все', value: 'all' },
    { title: 'Животное', value: 'animal' },
    { title: 'Человек', value: 'human' },
    { title: 'Природа', value: 'nature' }
  ];
}
```

**Mygallery.component.ts**

```
import { Component, OnInit, Input } from '@angular/core';
import { ImageService } from 'src/app/shared/services/image.service';
import { Imagee } from 'src/app/shared/model/image.model';
import { UserService } from 'src/app/shared/services/user.service';
import { User } from 'src/app/shared/model/user.model';

@Component({
  selector: 'app-mygallery',
  templateUrl: './mygallery.component.html',
  styleUrls: ['./mygallery.component.scss'],
})
export class MygalleryComponent implements OnInit {
  user: User;
  public visibleImages: Imagee[];
  // Create an input
  @Input() filterBy: string = 'all';

  constructor(private imageService: ImageService) {
    this.visibleImages=[];
    this.user = JSON.parse(window.localStorage.getItem('user') || '{}');
    this.imageService.getImagesUser(this.user.fio).subscribe((res) => {
      this.visibleImages = res;
    });
  }

  ngOnChanges() {
    this.imageService.getImagesUser(this.user.fio).subscribe((res) => {
      this.visibleImages = res;
    });
  }
```

```
  ngOnInit(): void {

  }
  concatUrl(url: string) {
    return `http://localhost:3002/${url}`;
  }
  currentFilter = { title: 'Все', value: 'all' };
  filters = [
    { title: 'Все', value: 'all' },
    { title: 'Животное', value: 'animal' },
    { title: 'Человек', value: 'human' },
    { title: 'Природа', value: 'nature' }
  ];
}
```

Personal.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ImageService } from 'src/app/shared/services/image.service';
import { Imagee } from 'src/app/shared/model/image.model';
import { UserService } from 'src/app/shared/services/user.service';
import { User } from 'src/app/shared/model/user.model';
@Component({
  selector: 'app-personal',
  templateUrl: './personal.component.html',
  styleUrls: ['./personal.component.scss']
})
export class PersonalComponent implements OnInit {
  user: User;

  constructor() {

    this.user = JSON.parse(window.localStorage.getItem('user')||'{}');


  }
  ngOnInit(): void {
  }


}
```

**Project.component.ts**

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-project',
  templateUrl: './project.component.html',
  styleUrls: ['./project.component.scss']
})
export class ProjectComponent implements OnInit {

  constructor() { }
```

```
  ngOnInit(): void {
  }

}
```

System.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import {SharedModule} from "../shared/shared.module";
import { SystemRoutingModule } from './system-routing.module';
import { ProjectComponent } from './project/project.component';
import { FooterComponent } from './shared/components/footer/footer.component';
import { HeaderComponent } from './shared/components/header/header.component';
import { EditComponent } from './edit/edit.component';
import { AddpictureComponent } from './addpicture/addpicture.component';
import { GalleryComponent } from './gallery/gallery.component';
import { MygalleryComponent } from './mygallery/mygallery.component';
import { PersonalComponent } from './personal/personal.component';
import { DropdownDirective } from './shared/directive/dropdown.directive';
import { EditService } from '../shared/services/edit.service';
import { ImageComponent } from './shared/components/image/image.component';
import { ImageService } from '../shared/services/image.service';
import { ImageFilterPipe } from '../shared/services/filter.pipe';


@NgModule({
  declarations: [
    ProjectComponent,
    FooterComponent,
    HeaderComponent,
    EditComponent,
    AddpictureComponent,
    GalleryComponent,
    MygalleryComponent,
    PersonalComponent,
    DropdownDirective,
    ImageComponent,
    ImageFilterPipe
  ],
  imports: [
    CommonModule,
    SystemRoutingModule,
    SharedModule,
  ],
  exports: [
    HeaderComponent,
    FooterComponent,
],
providers:[EditService,ImageService, ImageFilterPipe],
})
export class SystemModule { }
```