**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**По лабораторной работе № 3**

**по дисциплине «Web-Технологии»**

Тема: **МОДУЛЬ АДМИНИСТРИРОВАНИЯ ПРИЛОЖЕНИЯ «СОЦИАЛЬНАЯ СЕТЬ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0303 |  | Болкунов В.О. |
| Преподаватель |  | Беляев С. А. |

Санкт-Петербург

2022

**Цель работы.**

Целью работы является изучение возможностей применения компилятора Babel, библиотеки jQuery, препроцессора LESS, препроцессора SASS/SCSS, инструмента выполнения повторяющихся задач GULP, освоение инструмента сборки Webpack, регистрация разработанных модулей, формирования навыков построения структурированных web-приложений, освоение особенностей стандартных библиотек. Для достижения поставленной цели требуется решить следующие задачи:

– разработка интерфейса web-приложения с использованием Figma (<https://www.figma.com/>);

– создание web-сервера на основе express, настройка маршрутов, подготовка и обработка REST-запросов (серверная часть);

– разработка стилей web-приложения с использованием LESS или SASS/SCSS;

**Общая формулировка задачи.**

Необходимо создать web-приложение, обеспечивающее администрирование социальной сети: можно управлять участниками, их ролями, сообщениями. Основные требования следующие:

1. Перечень участников, их друзей, сообщений и т.п. хранится в JSONфайлах на сервере.

2. В качестве сервера используется Node.JS с модулем express. 3. Разработка ведется с использованием стандарта не ниже ECMAScript2015, используются ES6 модули.

4. Стили описываются с использованием LESS или SASS, при этом используются ключевые методы LESS/SASS (переменные, вложенные блоки, миксины, операторы и т. п.).

5. Клиентская часть разрабатывается с использованием jQuery (работа с DOM, AJAX-запросы), используются компоненты jQuery UI или Bootstrap.

6. Предусмотрена HTML-страница для списка пользователей (ФИО, дата рождения, email, фотография, роль, статус). Предусмотрена возможность редактировать данные пользователя, изменять роль (администратор, пользователь), изменять статус (не подтверждённый пользователь, активный, заблокированный).

7. Предусмотрены: – HTML-страница для списка друзей пользователя; – HTML-страница для списка новостей друзей пользователей.

8. Взаимодействие браузера с сервером осуществляется по протоколу HTTPS, все изменения сохраняются в соответствующие json-файлы на сервере.

9. Сборка клиентской части (преобразования less или sass, pug или ejs, babel, минификация) осуществляется с использованием двух инструментов: GULP и Webpack. Это должны быть две отдельные сборки в разные папки.

10. Регистрация и удаление разработанных модулей в npm.

11. Для всех страниц web-приложения разработан макет интерфейса с использованием Figma (https://www.figma.com/). Преимуществом будет, если будет предусмотрена работа с – фотографиями пользователя: список фотографий и возможность блокирования и активации фотографий администратором; – новостями: возможность блокирования и активации новостей.

**Выполнение работы.**

**Промежуточное api**

В модуле *api* описаны общие для клиентского и серверного приложений структуры данных и объекты.

* Роль пользователя

export enum Role {  
 *USER*,  
 *ADMIN*,  
}

* Статус пользователя/ресурса

export enum Status {  
 *ACTIVE*,  
 *UNCONFIRMED*,  
 *BLOCKED*,  
}

* Интерфейс индексируемых объектов

export interface Indexed {  
 id: number;  
}

* Интерфейс данных статуса

export interface StatusData {  
 status: Status;  
}

* Интерфейй статусных данных пользователя (его роль и статус)

export interface UserStatusData extends StatusData {  
 role: Role;  
}

* Интерфейс персональных данных

export interface PersonalData {  
 surname: string;  
 name: string;  
 lastName: string;  
 email: string;  
 birthDate: string;  
}

* Интерфейс данных пользователя (участвует в запросах)

export interface UserData extends Indexed, UserStatusData, PersonalData {  
 friends: number[];  
 photoId: number | null;  
}

* Интерфейс пользователя с паролем (хранится на сервере)

export interface User extends UserData {  
 password: string;  
}

* Интерфейс ресурса

export interface Resource extends Indexed, StatusData {  
 userId: number;  
 time: string;  
}

* Фотографии пользователя

export interface Photo extends Resource {  
 file: string;  
}

* Пост (новость) пользователя

export interface Post extends Resource {  
 text: string;  
 photosId: number[];  
}

* Авторизационные данные

export interface UserAuthData {  
 email: string;  
 password: string;  
}

* Объект с конфигурацией клиент-серверного взаимодействия

export const config

**Серверная часть (REST API)**

Управление пользователями и их ресурсами осуществляет класс Controller. export class Controller   
 private users: User[] = [];  
 private photos: Photo[] = [];  
 private posts: Post[] = [];  
 constructor(users: User[], photos: Photo[], posts: Post[])  
 authorize(data: UserAuthData): User | null   
 updateUserStatus(id: number, stat: UserStatusData)   
 updatePersonal(id: number, stat: PersonalData)   
 private getUser(id: number): User | null   
 getUsers(): number[]  
 getUserData(id: number): UserData | null   
 getFriends(id: number): number[] | null   
 private getItems<T extends Resource>(userId: number, items: T[]): number[]  
 private getItem<T extends Resource>(id: number, items: T[]): T | null   
 getPosts(userId: number): number[]  
 getPost(id: number): Post | null   
 getPhotos(userId: number): number[]  
 getPhoto(id: number): Photo | null   
 getFriendsPosts(id: number): number[]  
 private updateResourceStatus(res: Resource | null, sd: StatusData)   
 updatePostStatus(id: number, sd: StatusData)   
 updatePhotoStatus(id: number, sd: StatusData)

REST-api приложения реализовано с помощью роутера express в модуле *router.ts*

export function createAdminRouter(  
 controller: Controller,  
 storagePath: string  
): Router {  
 const router = Router();router.get("/users", (req, res) => {…});router.use("/user/:id", (req, res, next) => {…});router.use(  
 "/user/:id",  
 Router()  
 .get("/photos", (req, res) => {…})  
 .get("/posts", (req, res) => {…})  
 .get("/friends", (req, res) => {…})  
 .get("/friendsposts", (req, res) => {…})  
 );  
router  
 .route("/post/:postid")  
 .all((req, res, next) => {…})  
 .get((req, res) => {…})  
 .patch((req, res) => {…});router  
 .route("/photo/:photoid")  
 .all((req, res, next) => {…})  
 .get((req, res) => {…});router  
 .route("/photoinfo/:photoid")  
 .all((req, res, next) => {…})  
 .get((req, res) => {…})  
 .patch((req, res) => {…});router  
 .route("/user/:id")  
 .get((req, res) => {…})  
 .put((req, res) => {…})  
 .patch((req, res) => {…});  
 return router;  
}

Аутентификация пользователя осуществляется с помощью библиотеки passport.js с использованием стратегии Web-токенов. При отправке post-запроса с корректными данными **администратора,** пользователю отправляется cookie с подписанным токеном. Модуль *auth.ts* предоставляет middlware для аутентификации. Для входа требуется email и пароль администратора.

export const secretKey = "#S3cR3t\_K3y#";const cookieExtractor: JwtFromRequestFunction = function (req): string {…};  
export { passport };  
export function createAuthRouter(controller: Controller): Router {  
 const authRouter = Router();passport.use(  
 new JwtStrategy(  
 {  
 jwtFromRequest: cookieExtractor,  
 secretOrKey: secretKey,  
 },  
 (payload: Indexed, done) => {…}  
 )  
 );authRouter.post("/login", (req, res) => {…});authRouter.get(  
 "/login",  
 passport.authenticate("jwt", { session: false }),  
 (req, res) => {…}  
 );  
 return authRouter;  
}

**Клиентская часть**

Был создан макет основных страниц приложения, который представлен в файле *Lab3.fig****.***

Загрузку ресурсов осуществляют промежуточные классы в модуле *loaders*

* Интерфейс загрузчика

export interface ILoader   
 get url(): URL;  
 fetch(): void;

* Интерфейс загрузчика индексируемых объектов

export interface IndexedLoader   
 readonly id: number;  
 get endpoint(): string;

* Абстрактный класс загрузчика списков ресуров

export abstract class ListLoader implements ILoader   
 readonly list: number[] = [];  
 abstract get url(): URL;  
 async fetch()

* Загрузчик списка пользователей

export class UsersListLoader extends ListLoader   
 get url(): URL

* Абстрактный загрузчик списков данных ресурсов пользователя

abstract class UserDataListLoader extends ListLoader implements IndexedLoader   
 readonly id: number;  
 constructor(id: number)   
 abstract get endpoint(): string;  
 get url(): URL

* Загрузчик списка постов пользователя

export class PostsLoader extends UserDataListLoader   
 get endpoint(): string

* Загрузчик списка фотографий пользователя

export class PhotosLoader extends UserDataListLoader   
 get endpoint(): string

* Загрузчик списка друзей пользователя

export class FriendsLoader extends UserDataListLoader   
 get endpoint(): string

* Загрузчик списка постов друзей

export class FriendsPostsLoader extends PostsLoader   
 get endpoint(): string

* Абстрактный загрузчик объектов

export abstract class ObjectLoader<T> implements ILoader, IndexedLoader   
 readonly id: number;  
 data: T | null = null;  
 constructor(id: number)   
 abstract get endpoint(): string;  
 get url(): URL   
 async fetch()

* Загрузчик данных пользователя

export class UserLoader extends ObjectLoader<UserData>   
 get endpoint(): string   
 get avatarUrl(): string   
 get fullName()  
 updateStatus()  
 updatePersonal()

* Абстрактный загрузчик ресурсов

abstract class ResourceLoader<T extends Resource> extends ObjectLoader<T>   
 updateStatus(s: Status)

* Загрузчик поста

export class PostLoader extends ResourceLoader<Post>   
 get endpoint(): string

* Загрузчик фотографии (достаточно использовать url)

export class PhotoLoader extends ObjectLoader<Blob>   
 get endpoint(): string

* Загрузчик информации о фотографии

export class PhotoInfoLoader extends ResourceLoader<Photo>   
 get endpoint(): string

**Компоненты интерфейса**

* Миксин для компонентов списка, содержит список чисел (id объектов)

в качестве свойства (prop)

export default class List extends Vue   
 @Prop({ required: true }) readonly list!: number[];

* Компонент списка фотографий пользователя

export default class PhotosList extends List

* Компонент списка постов пользователя

export default class PostsList extends List

* Компонент списка пользователей

export default class UsersList extends List

* Компонент переключателя блокировки ресурса (поста и фотографий)

export default class BanSwitcher extends Vue   
 private Status = Status;  
 @Prop({ required: true }) status!: Status;  
 @Emit("switched")  
 private switchStatus(s: Status): Status

* Компонент карусели фотографий прилагаемых к посту

export default class Carousel extends Vue   
 @Prop({ required: true }) readonly list!: number[];  
 private id = 0;  
  
 $refs!: {  
 indicators: HTMLButtonElement[];  
 images: HTMLDivElement[];  
 };  
 private mounted()

* Компонент фотографии пользователя

export default class Photo extends Vue   
 private Status = Status;  
 @InjectReactive() readonly toaster!: Toaster | null;  
 @Prop({ required: true }) readonly id!: number;  
 private photo: PhotoLoader | null = null;  
 private photoInfo: PhotoInfoLoader | null = null;  
 private async mounted()  
 private switchStatus(s: Status)

* Компонент новости пользователя

export default class Post extends Vue   
 private Status = Status;  
 @InjectReactive() readonly toaster!: Toaster | null;  
 @Prop({ type: *Number*, required: true }) readonly id!: number;  
 private post: PostLoader | null = null;  
 private user: UserLoader | null = null;  
 private async mounted()  
 private switchStatus(s: Status)

* Компонент модального окна редактирования пользователя, позволяет

редактировать персональную информацию, статус и роль пользователя.

export default class RedactModal extends Vue   
 private Status = Status;  
 private Role = Role;  
 private modal!: Modal;  
 private loader: UserLoader | null = null;  
 private buffer: UserData | null = null;  
 $refs!: {  
 form: HTMLFormElement;  
 };  
 private mounted()  
 show(loader: UserLoader)   
 private submit()

* Компонент всплывающих сообщений

export default class Toaster extends Vue   
 private Status = Status;  
 @Prop({ default: 1500 }) interval!: number;  
 private msg = "";  
 private status: Status = Status.*UNCONFIRMED*;  
 private visible = false;  
 private timerId = 0;  
 show(status: Status, msg: string)

* Компонент карточки пользователя. В ней можно перейти к странице

пользователя, либо вызвать окно настроек пользователя.

export default class UserCard extends Vue   
 private Views = Views;  
 @InjectReactive() readonly redact!: RedactModal | null;  
 @Prop({ type: Number, required: true }) readonly id!: number;  
 private loader: UserLoader | null = null;  
 private async mounted()  
 private settings()

* Компонент иконки пользователя

export default class UserThumbnail extends Vue   
 private Views = Views;  
 @Prop({ required: true }) readonly loader!: UserLoader;

* Главный компонент приложения, предоставляет инъекцию

компонентов для модального окна формы и всплывающих сообщений.

export default class App extends Vue   
 private Views = Views;  
 $refs!: {  
 toaster: Toaster;  
 redact: RedactModal;  
 };  
 private mounted()  
 @ProvideReactive() toaster: Toaster | null = null;  
 @ProvideReactive() redact: RedactModal | null = null;

**Компоненты представления роутера**

* Страница авторизации пользователя, неавторизованный пользователь

переадресовывается на неё.

export default class LoginView extends Vue   
 @InjectReactive() readonly toaster!: Toaster | null;  
 private user: UserAuthData = { email: "", password: "" };  
 private valid = true;  
 private readonly url = new URL(  
 config.endpoints.login,  
 config.server  
 ).toString();  
 private async submit()  
 private reset()

* Страница списка всех пользователей

export default class UsersListView extends Vue   
 private users: UsersListLoader = new UsersListLoader();  
 private mounted()

* Страница пользователя, содержит в себе разделы навигации: новости,

фото, друзья, новости друзей

export default class UserView extends Vue   
 private id: number = *parseInt*(this.$route.params.user\_id);  
 private loader: ListLoader | null = null;  
  
 readonly categories = [  
 [Views.*POSTS*, "Новости"],  
 [Views.*PHOTOS*, "Фото"],  
 [Views.*FRIENDS*, "Друзья"],  
 [Views.*FRIENDSPOSTS*, "Новости друзей"],  
 ];  
 private mounted()  
 @Watch("$route")  
 onRouteChange(to: Route)

**Примеры экранных форм**.

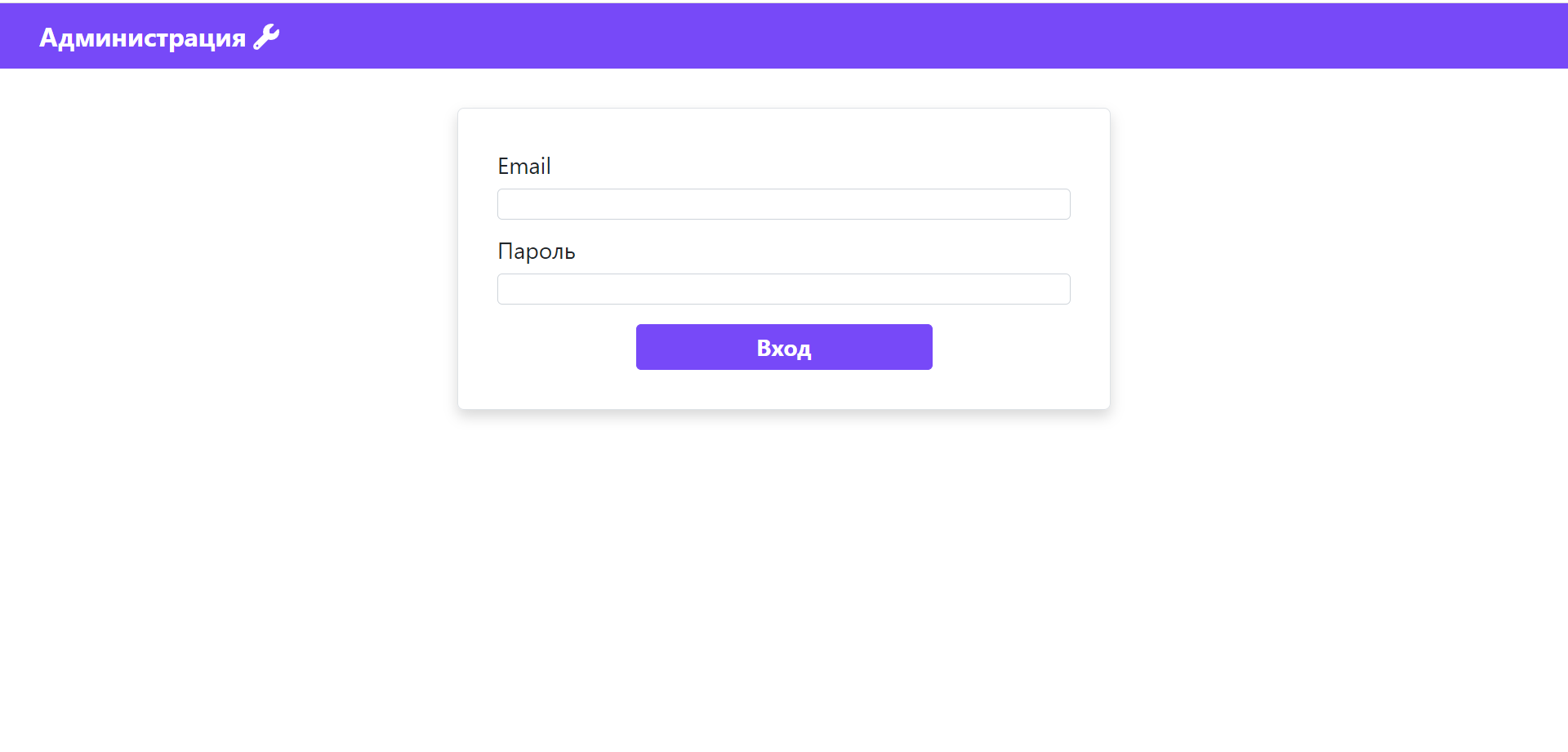


Рисунок 1: Стартовая страница

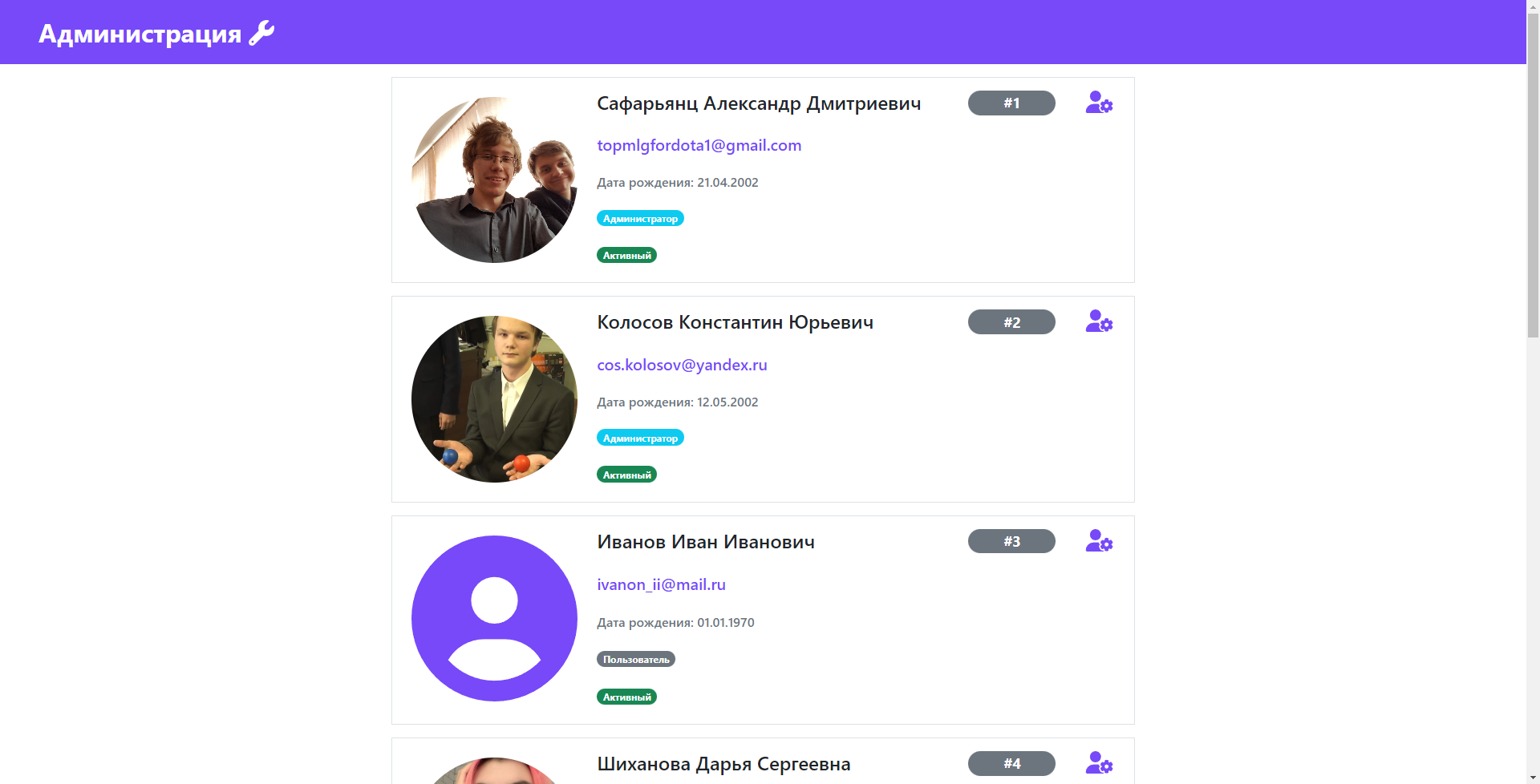


Рисунок 2: Список пользователей

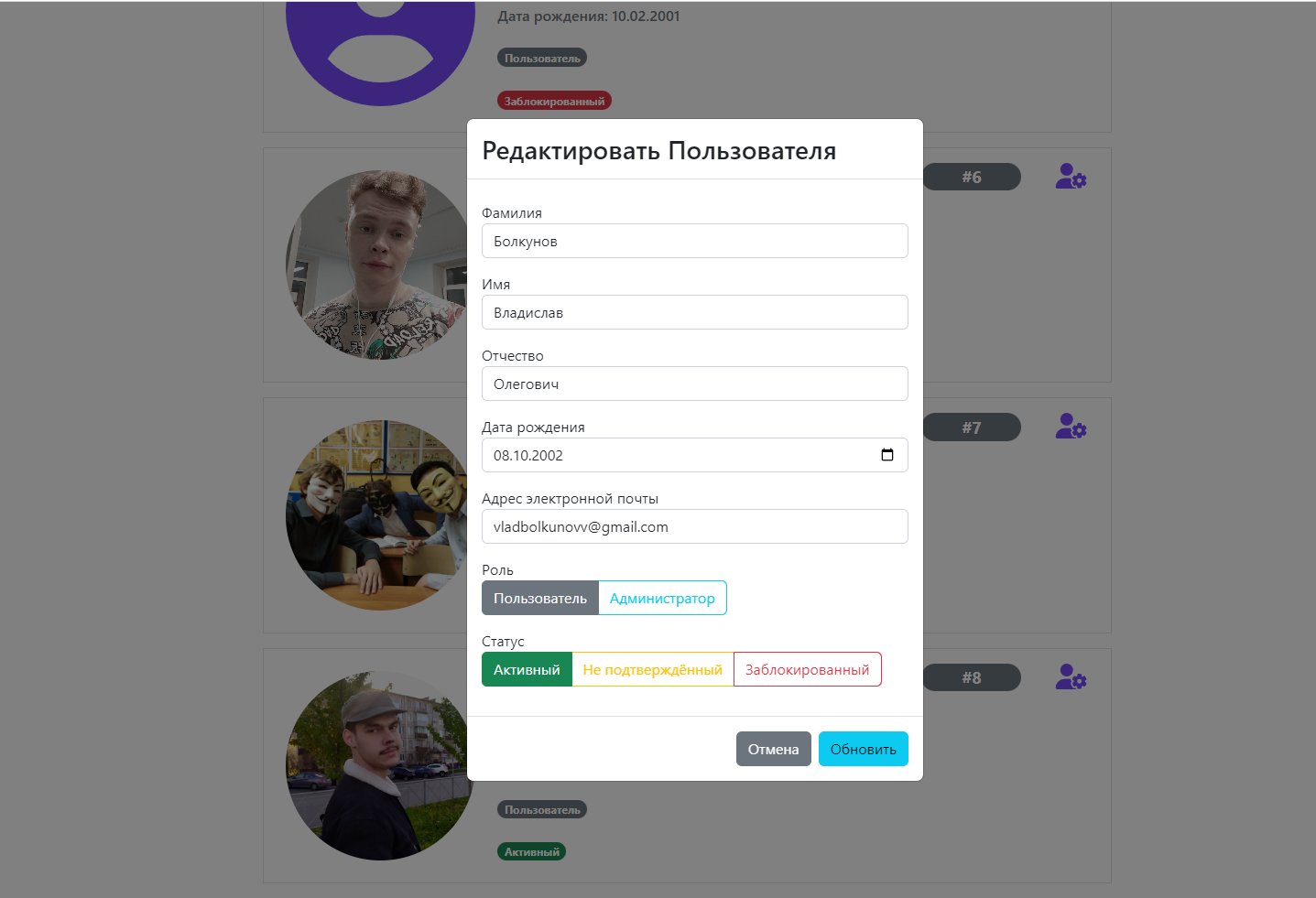


Рисунок 3: Страница редактирования пользователя

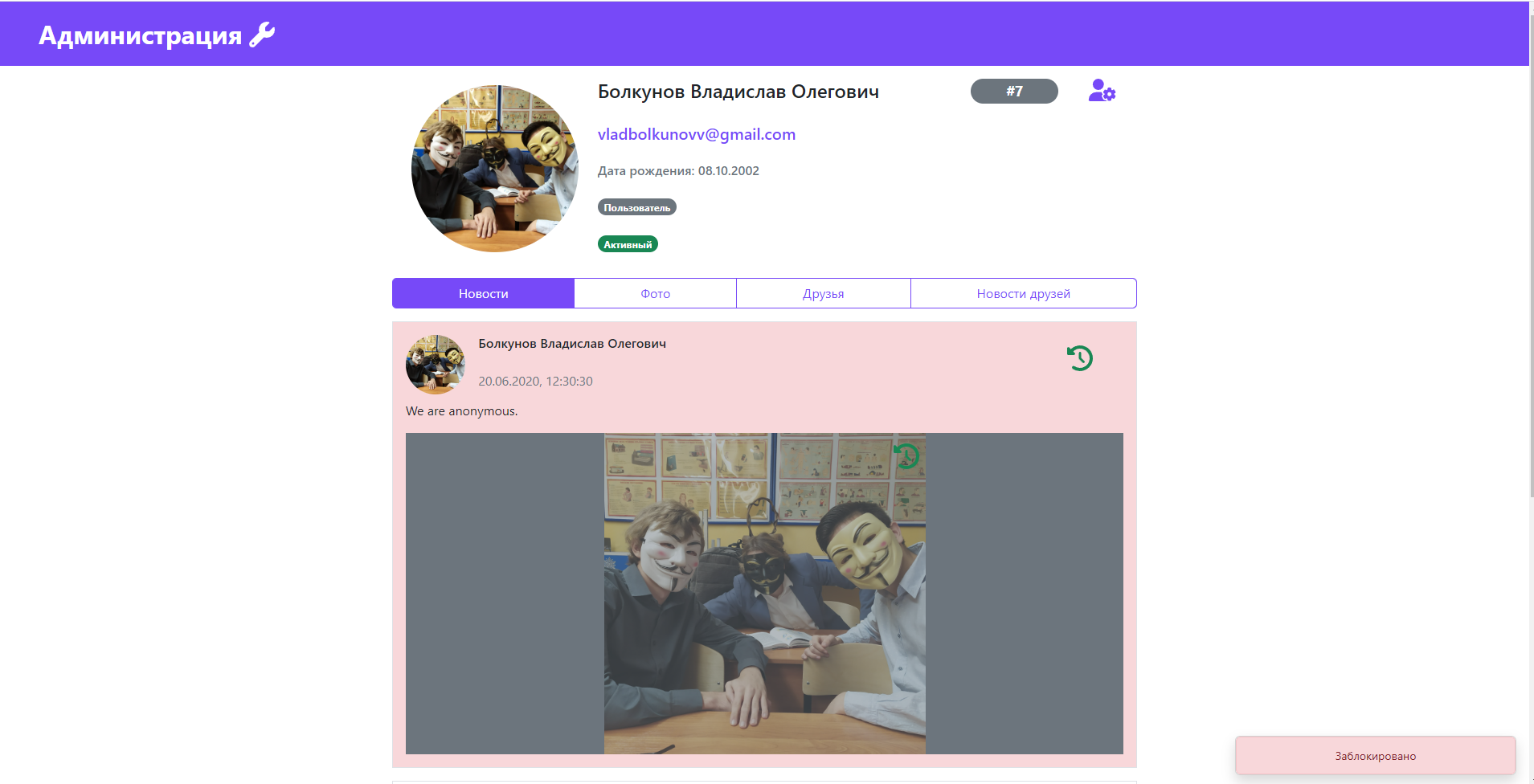


Рисунок 4: Новости пользователя

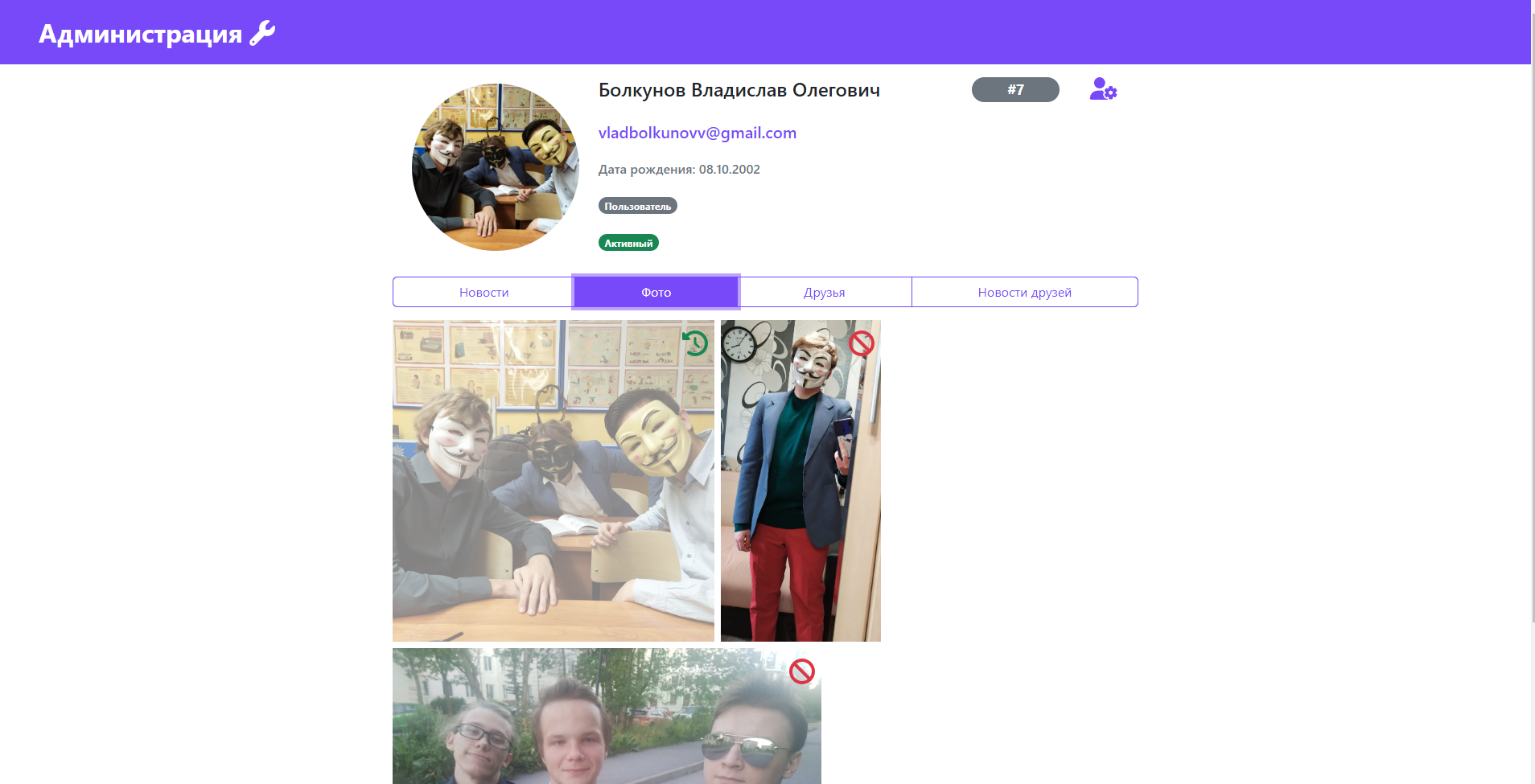


Рисунок 5: фотографии пользователя

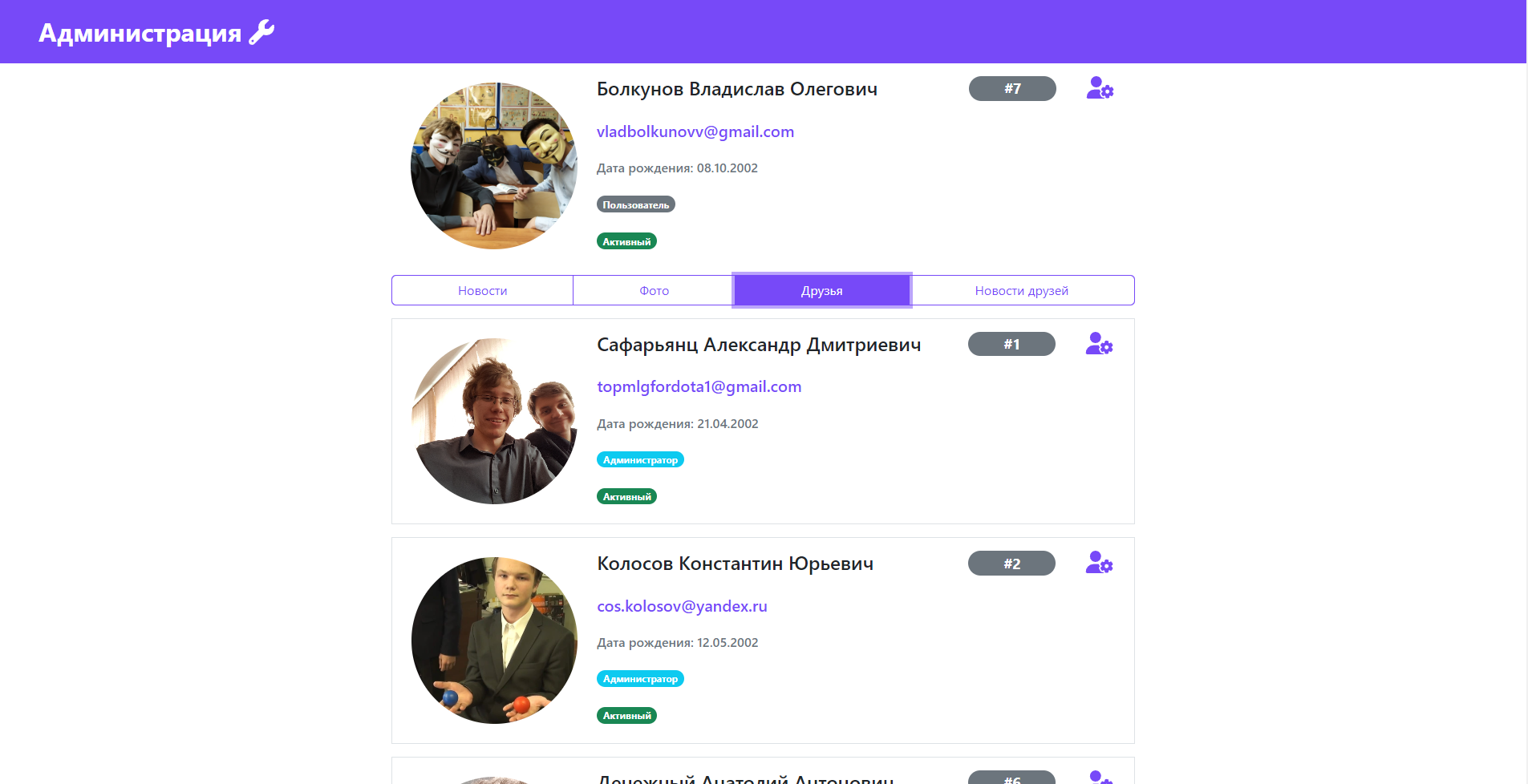
****

Рисунок 6: список друзей пользователя

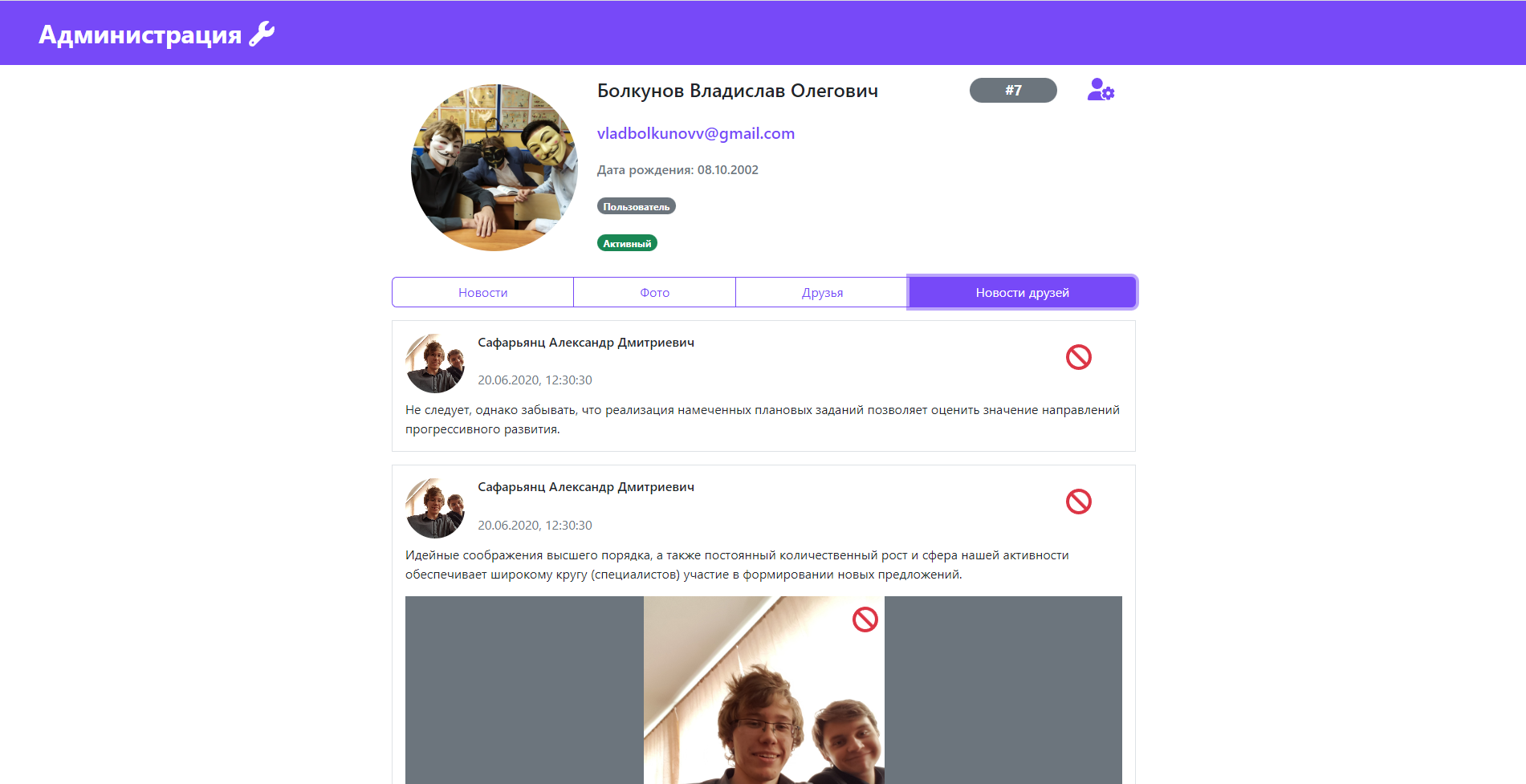
****

Рисунок 7: новости друзей пользователя

**Сборка**

Сборка клиентской части осуществляется и использованием плагина babel для vue-cli, что должно обеспечивать корректную работу приложения на (относительно) старых версиях браузеров.

**Защищённое соедниение**

Реализация защищённого соединения происходит с помощью веб-сервера nginx, который перенаправляет запросы с http протокола на https. Nginx также осуществляет проксирование запросов к API серверного приложения, что избавляет от необходимости добавлять ssl сертификаты в серверную часть приложения.

**Публикация пакета**

Созданное приложение было успешно опубликовано в реестр npm

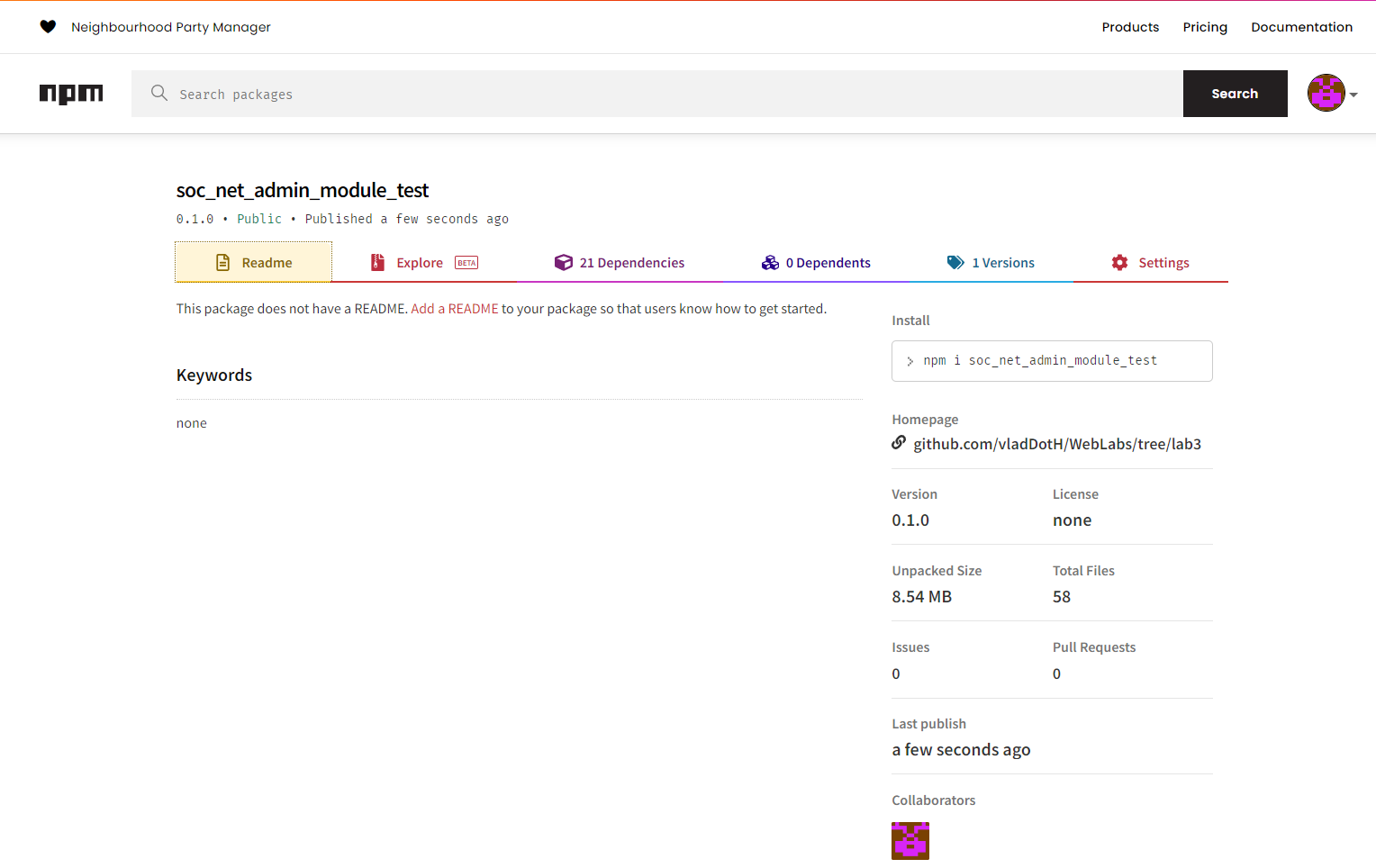


Рисунок 8: публикация пакета

После чего опубликованный модуль был удалён.

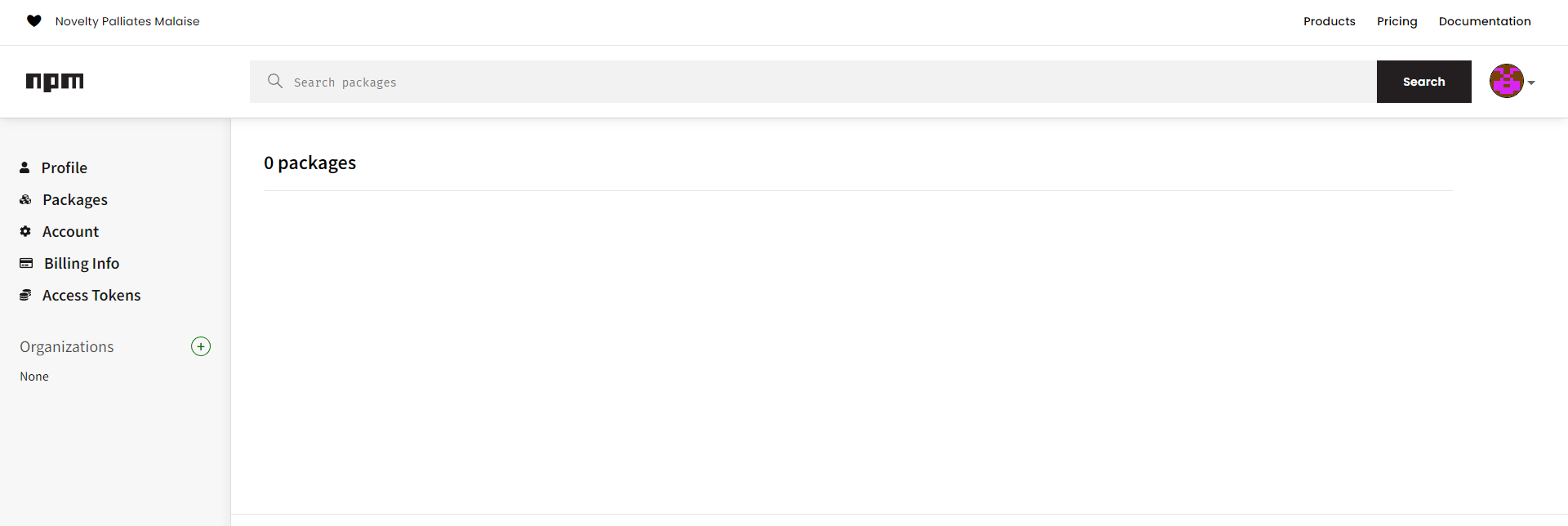


Рисунок 9: удаление пакета

**Выводы.**

В ходе было создано веб приложение администрирования социальной сети, с использованием REST архитектуры. Для соответствующего серверного приложения на основе созданного макета было реализовано клиентское приложение, взаимодействующее с сервером посредством ajax-запросов.

**ПРИЛОЖЕНИЕ А**

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

***Файл api/index.ts***

*// Роли пользователей*export enum Role {  
 *USER*,  
 *ADMIN*,  
}  
  
*// Статус пользователя / ресурса*export enum Status {  
 *ACTIVE*,  
 *UNCONFIRMED*,  
 *BLOCKED*,  
}  
  
*// Индексируемый объект*export interface Indexed {  
 id: number;  
}  
  
*// Объект со статусом*export interface StatusData {  
 status: Status;  
}  
  
*// Статусные данные пользователя*export interface UserStatusData extends StatusData {  
 role: Role;  
}  
  
*// Персональные данные пользователя*export interface PersonalData {  
 surname: string;  
 name: string;  
 lastName: string;  
 email: string;  
 birthDate: string;  
}  
  
*// Данные пользователя соц. сети*export interface UserData extends Indexed, UserStatusData, PersonalData {  
 friends: number[];  
 photoId: number | null;  
}  
  
*// Объект пользователя на сервере*export interface User extends UserData {  
 password: string;  
}  
  
*// Объект ресурса (фотография/пост)*export interface Resource extends Indexed, StatusData {  
 userId: number;  
 time: string;  
}  
  
export interface Photo extends Resource {  
 file: string;  
}  
  
export interface Post extends Resource {  
 text: string;  
 photosId: number[];  
}  
  
*// Авторизационные данные*export interface UserAuthData {  
 email: string;  
 password: string;  
}  
  
*// Настройки клиент-серверного взаимодействия*export const config = {  
 serverHost: "localhost",  
 server: "http://localhost:3000/api/",  
 client: "http://localhost:8080",  
 endpoints: {  
 login: "login/",  
 usersList: "users/",  
 user: "user/",  
 avatar: "avatar/",  
 postsList: "posts/",  
 post: "post/",  
 photosList: "photos/",  
 photo: "photo/",  
 photoInfo: "photoinfo/",  
 friends: "friends/",  
 friendsPosts: "friendsposts/",  
 },  
};

***Файл server/Controller.ts***

import {  
 PersonalData,  
 Photo,  
 Post,  
 UserStatusData,  
 User,  
 UserData,  
 Resource,  
 StatusData,  
 UserAuthData,  
} from "../api";  
  
*// Логика соц сети*export class Controller {  
 private users: User[] = [];  
 private photos: Photo[] = [];  
 private posts: Post[] = [];  
  
 constructor(users: User[], photos: Photo[], posts: Post[]) {  
 this.users = users;  
 this.photos = photos;  
 this.posts = posts;  
 }  
  
 authorize(data: UserAuthData): User | null {  
 const user = this.users.find((user) => user.email == data.email);  
 if (user && user.password == data.password) return user;  
 return null;  
 }  
  
 updateUserStatus(id: number, stat: UserStatusData) {  
 const user = this.getUser(id);  
 if (user) [user.status, user.role] = [stat.status, stat.role];  
 }  
  
 updatePersonal(id: number, stat: PersonalData) {  
 const user = this.getUser(id);  
 if (user)  
 [user.surname, user.name, user.lastName, user.email, user.birthDate] = [  
 stat.surname,  
 stat.name,  
 stat.lastName,  
 stat.email,  
 stat.birthDate,  
 ];  
 }  
  
 private getUser(id: number): User | null {  
 return this.users.find((user) => user.id == id) ?? null;  
 }  
  
 getUsers(): number[] {  
 return this.users.map((user) => user.id);  
 }  
  
 getUserData(id: number): UserData | null {  
 const user = this.getUser(id);  
 return user  
 ? ({  
 ...user,  
 password: undefined,  
 } as UserData)  
 : null;  
 }  
  
 getFriends(id: number): number[] | null {  
 return this.getUser(id)?.friends ?? null;  
 }  
  
 private getItems<T extends Resource>(userId: number, items: T[]): number[] {  
 return items.filter((item) => item.userId == userId).map((item) => item.id);  
 }  
  
 private getItem<T extends Resource>(id: number, items: T[]): T | null {  
 return items.find((item) => item.id == id) ?? null;  
 }  
  
 getPosts(userId: number): number[] {  
 return this.getItems(userId, this.posts);  
 }  
  
 getPost(id: number): Post | null {  
 return this.getItem(id, this.posts);  
 }  
  
 getPhotos(userId: number): number[] {  
 return this.getItems(userId, this.photos);  
 }  
  
 getPhoto(id: number): Photo | null {  
 return this.getItem(id, this.photos);  
 }  
  
 getFriendsPosts(id: number): number[] {  
 return (  
 this.getUser(id)  
 ?.friends.map((user) => this.getPosts(user))  
 .flat() ?? []  
 );  
 }  
  
 private updateResourceStatus(res: Resource | null, sd: StatusData) {  
 if (res) res.status = sd.status;  
 }  
  
 updatePostStatus(id: number, sd: StatusData) {  
 this.updateResourceStatus(this.getPost(id), sd);  
 }  
  
 updatePhotoStatus(id: number, sd: StatusData) {  
 this.updateResourceStatus(this.getPhoto(id), sd);  
 }  
}

***Файл server/router.ts***

import { Router } from "express";  
import path from "path";  
import { Controller } from "./Controller";  
import { PersonalData, StatusData, UserStatusData } from "../api";  
  
export function createAdminRouter(  
 controller: Controller,  
 storagePath: string  
): Router {  
 const router = Router();  
  
 *// Список пользователей* router.get("/users", (req, res) => {  
 res.json(controller.getUsers());  
 });  
  
 *// Middleware для извлечения id пользователя* router.use("/user/:id", (req, res, next) => {  
 res.locals.id = parseInt(req.params.id);  
 next();  
 });  
  
 *// Маршруты ресурсов пользователя* router.use(  
 "/user/:id",  
 Router()  
 .get("/photos", (req, res) => {  
 res.json(controller.getPhotos(res.locals.id));  
 })  
 .get("/posts", (req, res) => {  
 res.json(controller.getPosts(res.locals.id));  
 })  
 .get("/friends", (req, res) => {  
 res.json(controller.getFriends(res.locals.id));  
 })  
 .get("/friendsposts", (req, res) => {  
 res.json(controller.getFriendsPosts(res.locals.id));  
 })  
 );  
  
 *// Обращения к посту* router  
 .route("/post/:postid")  
 .all((req, res, next) => {  
 res.locals.id = parseInt(req.params.postid);  
 next();  
 })  
 .get((req, res) => {  
 res.json(controller.getPost(res.locals.id));  
 })  
 .patch((req, res) => {  
 controller.updatePostStatus(res.locals.id, req.body as StatusData);  
 res.sendStatus(200);  
 });  
  
 *// Обращения к изображению* router  
 .route("/photo/:photoid")  
 .all((req, res, next) => {  
 res.locals.id = parseInt(req.params.photoid);  
 next();  
 })  
 .get((req, res) => {  
 const photo = controller.getPhoto(res.locals.id);  
 if (photo) res.download(path.resolve(storagePath, photo.file));  
 else res.sendStatus(204);  
 });  
  
 *// Обращение к информации фото* router  
 .route("/photoinfo/:photoid")  
 .all((req, res, next) => {  
 res.locals.id = parseInt(req.params.photoid);  
 next();  
 })  
 .get((req, res) => {  
 const photoid = parseInt(req.params.photoid);  
 res.json(controller.getPhoto(photoid));  
 })  
 .patch((req, res) => {  
 controller.updatePhotoStatus(res.locals.id, req.body as StatusData);  
 res.sendStatus(200);  
 });  
  
 *// Маршруты данных пользователя* router  
 .route("/user/:id")  
 .get((req, res) => {  
 res.json(controller.getUserData(res.locals.id));  
 })  
 .put((req, res) => {  
 const data = req.body as PersonalData;  
 controller.updatePersonal(res.locals.id, data);  
 res.sendStatus(200);  
 })  
 .patch((req, res) => {  
 const data = req.body as UserStatusData;  
 controller.updateUserStatus(res.locals.id, data);  
 res.sendStatus(200);  
 });  
  
 return router;  
}

***Файл server/auth.ts***

*// Закрытый ключ*export const secretKey = "#S3cR3t\_K3y#";  
  
*// Извлечение токена из куки*const cookieExtractor: JwtFromRequestFunction = function (req): string {  
 return req.signedCookies?.token ?? null;  
};  
  
export { passport };  
  
export function createAuthRouter(controller: Controller): Router {  
 const authRouter = Router();  
  
 *// Проверка токена и роли пользователя* passport.use(  
 new JwtStrategy(  
 {  
 jwtFromRequest: cookieExtractor,  
 secretOrKey: secretKey,  
 },  
 (payload: Indexed, done) => {  
 if (controller.getUserData(payload.id)?.role == Role.ADMIN)  
 done(null, { id: payload.id });  
 else done(null, false);  
 }  
 )  
 );  
  
 *// Вход по email и паролю, устанавливает токен в куки* authRouter.post("/login", (req, res) => {  
 const data = req.body as UserAuthData,  
 user = controller.authorize(data);  
 if (user?.role == Role.ADMIN) {  
 const token = jwt.sign({ id: user.id }, secretKey, {  
 expiresIn: "1h",  
 });  
 res.cookie("token", token, {  
 maxAge: 60 \* 60 \* 1000,  
 httpOnly: true,  
 signed: true,  
 });  
  
 res.sendStatus(200);  
 } else res.sendStatus(401);  
 });  
  
 *// Точка проверки действительности токена* authRouter.get(  
 "/login",  
 passport.authenticate("jwt", { session: false }),  
 (req, res) => {  
 res.sendStatus(200);  
 }  
 );  
  
 return authRouter;  
}

***Файл server/index.ts***

import express from "express";  
import cors from "cors";  
import { config } from "../api";  
import { createAuthRouter, passport, secretKey } from "./auth";  
import cookieParser from "cookie-parser";  
import { createAdminRouter } from "./router";  
import fs from "fs";  
import path from "path";  
import { Controller } from "./Controller";  
  
const app = express();  
const port = 3000;  
  
const storagePath = "./storage";  
  
function load<T>(file: string): T {  
 return JSON.parse(  
 fs.readFileSync(path.resolve(storagePath, file), "utf-8")  
 ) as T;  
}  
  
const controller = new Controller(  
 load("users.json"),  
 load("photos.json"),  
 load("posts.json")  
);  
  
app.use(  
 cors({  
 origin: config.client,  
 credentials: true,  
 })  
);  
app.use(cookieParser(secretKey));  
app.use(express.json());  
app.use(express.urlencoded({ extended: true }));  
app.use(passport.initialize());  
app.use("/api", createAuthRouter(controller));  
app.use(  
 "/api",  
 passport.authenticate("jwt", { session: false }),  
 createAdminRouter(controller, storagePath)  
);  
  
app.listen(port, config.serverHost, function () {  
 console.log(`Server started at http://${config.serverHost}:${port}`);  
});

***Файл src/loaders/base.ts***

*// Интерфейс загрузчика*export interface ILoader {  
 get url(): URL;  
 fetch(): void;  
}  
  
*// Интерфейс загрузчика индексируемых данных*export interface IndexedLoader {  
 readonly id: number;  
 get endpoint(): string;  
}

***Файл src/loaders/listLoaders.ts***

import { config } from "@/../api";  
import axios from "axios";  
import { ILoader, IndexedLoader } from "./base";  
  
*// Абстрактный загрузчик списка*export abstract class ListLoader implements ILoader {  
 readonly list: number[] = [];  
  
 abstract get url(): URL;  
  
 async fetch() {  
 if (this.url)  
 this.list.splice(  
 0,  
 this.list.length,  
 ...(await axios.get<number[]>(this.url.toString())).data  
 );  
 }  
}  
  
*// Загрузчик списка пользователей*export class UsersListLoader extends ListLoader {  
 get url(): URL {  
 return new URL(config.endpoints.usersList, config.server);  
 }  
}  
  
*// Абстрактный загрузчик списков данных пользователя*abstract class UserDataListLoader extends ListLoader implements IndexedLoader {  
 readonly id: number;  
  
 constructor(id: number) {  
 super();  
 this.id = id;  
 }  
  
 abstract get endpoint(): string;  
  
 get url(): URL {  
 return new URL(  
 config.endpoints.user + `${this.id}/` + this.endpoint,  
 config.server  
 );  
 }  
}  
  
*// Загрузчик списка постов пользователя*export class PostsLoader extends UserDataListLoader {  
 get endpoint(): string {  
 return config.endpoints.postsList;  
 }  
}  
  
*// Загрузчик списка фотографий пользователя*export class PhotosLoader extends UserDataListLoader {  
 get endpoint(): string {  
 return config.endpoints.photosList;  
 }  
}  
  
*// Загрузчик списка друзей пользователя*export class FriendsLoader extends UserDataListLoader {  
 get endpoint(): string {  
 return config.endpoints.friends;  
 }  
}  
  
*// Загрузчик постов друзей пользователя*export class FriendsPostsLoader extends PostsLoader {  
 get endpoint(): string {  
 return config.endpoints.friendsPosts;  
 }  
}

***Файл src/loaders/dataLoaders.ts***

import axios from "axios";  
import { ILoader, IndexedLoader } from "./base";  
import {  
 config,  
 PersonalData,  
 Photo,  
 Post,  
 Resource,  
 Status,  
 StatusData,  
 UserData,  
 UserStatusData,  
} from "@/../api";  
  
*// Абстрактный загрузчик объектов*export abstract class ObjectLoader<T> implements ILoader, IndexedLoader {  
 readonly id: number;  
 data: T | null = null;  
  
 constructor(id: number) {  
 this.id = id;  
 }  
  
 abstract get endpoint(): string;  
  
 get url(): URL {  
 return new URL(this.endpoint + `${this.id}/`, config.server);  
 }  
  
 async fetch() {  
 this.data = (await axios.get<T>(this.url.toString())).data;  
 }  
}  
  
*// Загрузчик пользователя*export class UserLoader extends ObjectLoader<UserData> {  
 get endpoint(): string {  
 return config.endpoints.user;  
 }  
  
 get avatarUrl(): string {  
 return new URL(  
 config.endpoints.photo + `${this.data?.photoId}`,  
 config.server  
 ).toString();  
 }  
  
 get fullName() {  
 return this.data  
 ? `${this.data.surname} ${this.data.name} ${this.data.lastName ?? ""}`  
 : "";  
 }  
  
 updateStatus() {  
 return axios.patch<UserStatusData>(this.url.toString(), {  
 status: this.data?.status,  
 role: this.data?.role,  
 } as UserStatusData);  
 }  
  
 updatePersonal() {  
 return axios.put<PersonalData>(this.url.toString(), {  
 name: this.data?.name,  
 surname: this.data?.surname,  
 lastName: this.data?.lastName,  
 birthDate: this.data?.birthDate,  
 email: this.data?.email,  
 } as PersonalData);  
 }  
}  
  
*// Загрузчик ресурсов*abstract class ResourceLoader<T extends Resource> extends ObjectLoader<T> {  
 updateStatus(s: Status) {  
 if (this.data) this.data.status = s;  
 return axios.patch<StatusData>(this.url.toString(), {  
 status: s,  
 } as StatusData);  
 }  
}  
  
*// Загрузчик поста*export class PostLoader extends ResourceLoader<Post> {  
 get endpoint(): string {  
 return config.endpoints.post;  
 }  
}  
  
*// Загрузчик фотографии (достаточно использовать сгенерированный url)*export class PhotoLoader extends ObjectLoader<Blob> {  
 get endpoint(): string {  
 return config.endpoints.photo;  
 }  
}  
  
*// Загрузчик информации о фотографии*export class PhotoInfoLoader extends ResourceLoader<Photo> {  
 get endpoint(): string {  
 return config.endpoints.photoInfo;  
 }  
}

***Файл src/loaders/index.ts***

export \* from "./listLoaders";  
export \* from "./dataLoaders";

***Файл src/router/index.ts***

import Vue from "vue";  
import VueRouter, { RouteConfig } from "vue-router";  
import axios from "axios";  
import { config } from "@/../api";  
  
Vue.use(VueRouter);  
  
export enum Views {  
 LOGIN = "login",  
 HOME = "home",  
 USER = "user",  
 POSTS = "posts",  
 PHOTOS = "photos",  
 FRIENDS = "friends",  
 FRIENDSPOSTS = "friends-posts",  
}  
  
const routes: Array<RouteConfig> = [  
 {  
 path: "/",  
 name: Views.LOGIN,  
 component: () => import("../views/LoginView.vue"),  
 },  
 {  
 path: "/home",  
 name: Views.HOME,  
 component: () => import("../views/UsersListView.vue"),  
 },  
 {  
 path: "/user/:user\_id",  
 name: Views.USER,  
 component: () => import("../views/UserView.vue"),  
 children: [  
 {  
 name: Views.POSTS,  
 path: "posts",  
 component: () => import("../components/lists/PostsList.vue"),  
 },  
 {  
 name: Views.PHOTOS,  
 path: "photos",  
 component: () => import("../components/lists/PhotosList.vue"),  
 },  
 {  
 name: Views.FRIENDS,  
 path: "friends",  
 component: () => import("../components/lists/UsersList.vue"),  
 },  
 {  
 name: Views.FRIENDSPOSTS,  
 path: "friends-posts",  
 component: () => import("../components/lists/PostsList.vue"),  
 },  
 ],  
 },  
];  
  
const router = new VueRouter({  
 routes,  
});  
  
*// Проверка авторизации пользователя*async function isAuthenticated(): Promise<boolean> {  
 try {  
 await axios.get(new URL(config.endpoints.login, config.server).toString(), {  
 withCredentials: true,  
 });  
 return true;  
 } catch (err) {  
 return false;  
 }  
}  
  
*// Перенаправление неавторизованного пользователя на страницу входа*router.beforeEach(async (to, from, next) => {  
 const auth = await isAuthenticated();  
 if (auth) {  
 if (to.name === Views.LOGIN) next({ name: Views.HOME });  
 else next();  
 } else {  
 if (to.name !== Views.LOGIN) next({ name: Views.LOGIN });  
 else next();  
 }  
});  
  
export default router;

***Файл styles/main.css***

$primary: #7749f8;  
@import "bootstrap/scss/bootstrap";

***Файл src/components/lists/List.ts***

import Vue from "vue";  
import Component from "vue-class-component";  
import { Prop } from "vue-property-decorator";  
  
*// Миксин списка объектов*@Component  
export default class List extends Vue {  
 @Prop({ required: true }) readonly list!: number[];  
}

***Файл src/components/lists/PhotosList.vue***

<template>  
 <section class="row gap-2 g-0">  
 <div v-for="photo of list" :key="photo" class="image">  
 <Photo :id="photo" class="w-100 h-100"> </Photo>  
 </div>  
 </section>  
</template>  
  
<script lang="ts">  
import { Component } from "vue-property-decorator";  
  
import Photo from "@/components/Photo.vue";  
import List from "@/components/lists/List";  
  
*// Список фотографий пользователя*@Component({  
 components: { Photo },  
})  
export default class PhotosList extends List {}  
</script>  
  
<style scoped lang="scss">  
.photos {  
 display: grid;  
 grid-template-columns: repeat(3, 1fr);  
}  
.image {  
 height: 400px;  
 width: auto;  
}  
</style>

***Файл src/components/lists/PostsList.vue***

<template>  
 <section class="m-auto">  
 <Post  
 v-for="post of list"  
 :key="post"  
 :id="post"  
 class="border mt-3"  
 ></Post>  
 </section>  
</template>  
  
<script lang="ts">  
import { Component } from "vue-property-decorator";  
import Post from "@/components/Post.vue";  
import List from "@/components/lists/List";  
  
*// Список постов*@Component({  
 components: { Post },  
})  
export default class PostsList extends List {}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/lists/UsersList.vue***

<template>  
 <section class="m-auto">  
 <div v-for="id of list" :key="id" class="border mt-3">  
 <UserCard :id="id"></UserCard>  
 </div>  
 </section>  
</template>  
  
<script lang="ts">  
import { Component } from "vue-property-decorator";  
import UserCard from "@/components/UserCard.vue";  
import List from "@/components/lists/List";  
  
*// Список пользователей*@Component({  
 components: { UserCard },  
})  
export default class UsersList extends List {}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/BanSwitcher.vue***

<template>  
 <div>  
 <font-awesome-icon  
 v-if="status === Status.ACTIVE"  
 icon="fa-solid fa-ban"  
 class="text-danger"  
 @click="switchStatus(Status.BLOCKED)"  
 role="button"  
 />  
 <font-awesome-icon  
 v-else  
 icon="fa-solid fa-clock-rotate-left"  
 class="text-success"  
 @click="switchStatus(Status.ACTIVE)"  
 role="button"  
 />  
 </div>  
</template>  
  
<script lang="ts">  
import { Component, Emit, Prop, Vue } from "vue-property-decorator";  
import { Status } from "@/../api";  
  
*// Компонент блокирования/разблокирования ресурсов*@Component({})  
export default class BanSwitcher extends Vue {  
 private Status = Status;  
 @Prop({ required: true }) status!: Status;  
  
 @Emit("switched")  
 private switchStatus(s: Status): Status {  
 return s;  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/Carusel.vue***

<template>  
 <section :id="'carousel' + id" class="carousel slide">  
 <div class="carousel-indicators" v-show="list.length > 1">  
 <button  
 v-for="(photo, index) of list"  
 :key="index"  
 type="button"  
 :data-bs-target="'#carousel' + id"  
 :data-bs-slide-to="index"  
 ref="indicators"  
 ></button>  
 </div>  
 <div class="carousel-inner">  
 <div  
 v-for="(photo, index) of list"  
 :key="index"  
 ref="images"  
 class="carousel-item w-100"  
 >  
 <div class="w-100 h-100 d-flex justify-content-center">  
 <Photo :id="photo" class=""></Photo>  
 </div>  
 </div>  
 </div>  
 </section>  
</template>  
  
<script lang="ts">  
import { Component, Prop, Vue } from "vue-property-decorator";  
import Photo from "@/components/Photo.vue";  
  
*// Карусель фотографий в посте*@Component({  
 components: { Photo },  
})  
export default class Carousel extends Vue {  
 @Prop({ required: true }) readonly list!: number[];  
 private id = 0;  
  
 $refs!: {  
 indicators: HTMLButtonElement[];  
 images: HTMLDivElement[];  
 };  
  
 private mounted() {  
 if (this.list.length) {  
 this.id = Date.now();  
 this.$refs.indicators[0].classList.add("active");  
 this.$refs.images[0].classList.add("active");  
 }  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "@/styles/main.scss";  
.carousel-item {  
 background: $gray-600;  
 height: 400px;  
 width: auto;  
}  
</style>

***Файл src/components/Photo.vue***

<template>  
 <div class="position-relative" v-if="photo && photoInfo?.data">  
 <img  
 :src="photo.url.toString()"  
 :alt="id"  
 class="h-100 w-100"  
 :class="{ banned: photoInfo.data.status === Status.BLOCKED }"  
 />  
 <BanSwitcher  
 :status="photoInfo.data.status"  
 class="position-absolute fs-2 pt-1 pe-2 icon"  
 @switched="switchStatus"  
 />  
 </div>  
</template>  
  
<script lang="ts">  
import { Component, InjectReactive, Prop, Vue } from "vue-property-decorator";  
import { PhotoInfoLoader, PhotoLoader } from "@/loaders";  
import { Status } from "@/../api";  
import BanSwitcher from "@/components/BanSwitcher.vue";  
import Toaster, { getBanMsg } from "@/components/Toaster.vue";  
  
*// Фотография в галерее/постах*@Component({  
 components: { BanSwitcher },  
})  
export default class Photo extends Vue {  
 private Status = Status;  
 @InjectReactive() readonly toaster!: Toaster | null;  
 @Prop({ required: true }) readonly id!: number;  
  
 private photo: PhotoLoader | null = null;  
 private photoInfo: PhotoInfoLoader | null = null;  
  
 private async mounted() {  
 this.photo = new PhotoLoader(this.id);  
 this.photoInfo = new PhotoInfoLoader(this.id);  
 await this.photoInfo?.fetch();  
 }  
  
 private switchStatus(s: Status) {  
 this.toaster?.show(s, getBanMsg(s));  
 this.photoInfo?.updateStatus(s);  
 }  
}  
</script>  
  
<style scoped lang="scss">  
.banned {  
 opacity: 0.5;  
}  
  
img {  
 object-fit: contain;  
 transition: all 0.3s ease-in-out;  
}  
  
.icon {  
 top: 0;  
 right: 0;  
}  
</style>

***Файл src/components/Post.vue***

<template>  
 <article  
 v-if="post?.data"  
 class="p-3 post"  
 :class="{ banned: post.data.status === Status.*BLOCKED* }"  
 >  
 <section v-if="user?.data" class="d-flex">  
 <div  
 class="col-2 col-md-1 d-flex align-items-center justify-content-center card-image"  
 >  
 <UserThumbnail :loader="user"></UserThumbnail>  
 </div>  
 <div  
 class="col-9 col-md-10 d-flex flex-column justify-content-between ps-3 card-content"  
 >  
 <h6>{{ user.fullName }}</h6>  
 <h6 class="text-muted fw-normal">  
 {{ new Date(post.data.time).toLocaleString() }}  
 </h6>  
 </div>  
 <BanSwitcher  
 :status="post.data.status"  
 class="col-1 text-danger icon fs-2 p-1"  
 @switched="switchStatus"  
 />  
 </section>  
 <section class="mt-2">  
 {{ post.data.text }}  
 </section>  
 <section>  
 <Carousel  
 v-if="post.data.photosId.length"  
 :list="post.data.photosId"  
 class="mt-3"  
 ></Carousel>  
 </section>  
 </article>  
</template>  
  
<script lang="ts">  
import { *Component*, *InjectReactive*, *Prop*, Vue } from "vue-property-decorator";  
import { PostLoader, UserLoader } from "@/loaders";  
import Carousel from "@/components/Carousel.vue";  
import UserThumbnail from "@/components/UserThumbnail.vue";  
import BanSwitcher from "@/components/BanSwitcher.vue";  
import Toaster, { *getBanMsg* } from "@/components/Toaster.vue";  
import { Status } from "@/../api";  
  
*// Компонент поста*@Component({  
 components: { BanSwitcher, UserThumbnail, Carousel },  
})  
export default class Post extends Vue {  
 private Status = Status;  
 @InjectReactive() readonly toaster!: Toaster | null;  
 @Prop({ type: *Number*, required: true }) readonly id!: number;  
  
 private post: PostLoader | null = null;  
 private user: UserLoader | null = null;  
  
 private async mounted() {  
 this.post = new PostLoader(this.id);  
 await this.post.fetch();  
 if (this.post.data) this.user = new UserLoader(this.post.data.userId);  
 this.user?.fetch();  
 }  
  
 private switchStatus(s: Status) {  
 this.toaster?.show(s, *getBanMsg*(s));  
 this.post?.updateStatus(s);  
 }  
}  
</script>  
  
<style scoped lang="scss">  
@import "@/styles/main.scss";  
  
.post {  
 transition: all 0.3s ease-in-out;  
}  
.banned {  
 background-color: $red-100;  
}  
</style>

***Файл src/components/RedactModal.vue***

<template>  
 <form  
 class="modal fade"  
 data-bs-backdrop="static"  
 ref="form"  
 @submit.prevent="submit"  
 >  
 <div class="modal-dialog modal-dialog-centered modal-md">  
 <div class="modal-content">  
 <h3 class="modal-header">Редактировать Пользователя</h3>  
 <div class="modal-body" v-if="loader?.data">  
 <div class="mb-3">  
 <label>Фамилия</label>  
 <input  
 required  
 class="form-control"  
 type="text"  
 v-model="buffer.surname"  
 placeholder="Иванов"  
 />  
 </div>  
  
 <div class="mb-3">  
 <label>Имя</label>  
 <input  
 required  
 class="form-control"  
 type="text"  
 v-model="buffer.name"  
 placeholder="Иван"  
 />  
 </div>  
  
 <div class="mb-3">  
 <label>Отчество</label>  
 <input  
 class="form-control"  
 type="text"  
 v-model="buffer.lastName"  
 placeholder="Иванович"  
 />  
 </div>  
  
 <div class="mb-3">  
 <label>Дата рождения</label>  
 <input  
 required  
 class="form-control"  
 type="date"  
 v-model="buffer.birthDate"  
 placeholder="1970-01-01"  
 />  
 </div>  
  
 <div class="mb-3">  
 <label>Адрес электронной почты</label>  
 <input  
 required  
 class="form-control"  
 type="email"  
 v-model="buffer.email"  
 placeholder="example@mail.com"  
 />  
 </div>  
  
 <div class="mb-3">  
 <label class="d-block">Роль</label>  
 <div class="btn-group">  
 <input  
 type="radio"  
 class="btn-check"  
 name="role"  
 id="user"  
 v-model="buffer.role"  
 :value="Role.*USER*"  
 />  
 <label class="btn btn-outline-secondary" for="user"  
 >Пользователь</label  
 >  
  
 <input  
 type="radio"  
 class="btn-check"  
 name="role"  
 id="admin"  
 v-model="buffer.role"  
 :value="Role.*ADMIN*"  
 />  
 <label class="btn btn-outline-info" for="admin"  
 >Администратор</label  
 >  
 </div>  
 </div>  
  
 <div class="mb-3">  
 <label class="d-block">Статус</label>  
 <div class="btn-group">  
 <input  
 type="radio"  
 class="btn-check"  
 name="status"  
 id="active"  
 v-model="buffer.status"  
 :value="Status.*ACTIVE*"  
 />  
 <label class="btn btn-outline-success" for="active"  
 >Активный</label  
 >  
  
 <input  
 type="radio"  
 class="btn-check"  
 name="status"  
 id="unconfirmed"  
 v-model="buffer.status"  
 :value="Status.*UNCONFIRMED*"  
 />  
 <label class="btn btn-outline-warning" for="unconfirmed">  
 Не подтверждённый  
 </label>  
  
 <input  
 type="radio"  
 class="btn-check"  
 name="status"  
 id="blocked"  
 v-model="buffer.status"  
 :value="Status.*BLOCKED*"  
 />  
 <label class="btn btn-outline-danger" for="blocked">  
 Заблокированный  
 </label>  
 </div>  
 </div>  
 </div>  
  
 <div class="modal-footer">  
 <button type="button" class="btn btn-secondary" @click="modal.hide()">  
 Отмена  
 </button>  
 <button  
 type="button"  
 class="btn btn-info"  
 @click="$refs.form.requestSubmit()"  
 >  
 Обновить  
 </button>  
 </div>  
 </div>  
 </div>  
 </form>  
</template>  
  
<script lang="ts">  
import { *Component*, Vue } from "vue-property-decorator";  
import { Modal } from "bootstrap";  
import { UserLoader } from "@/loaders";  
import { Role, Status, UserData } from "@/../api";  
  
*// Всплывающее окно редактирования пользователя*@Component  
export default class RedactModal extends Vue {  
 private Status = Status;  
 private Role = Role;  
 private modal!: Modal;  
 private loader: UserLoader | null = null;  
 private buffer: UserData | null = null;  
  
 $refs!: {  
 form: HTMLFormElement;  
 };  
  
 private mounted() {  
 this.modal = new Modal(this.$refs.form);  
 }  
  
 show(loader: UserLoader) {  
 this.loader = loader;  
 if (loader.data) this.buffer = { ...loader.data };  
 this.modal.show();  
 }  
  
 private submit() {  
 this.modal.hide();  
 if (this.loader?.data && this.buffer) this.loader.data = { ...this.buffer };  
 this.loader?.updatePersonal();  
 this.loader?.updateStatus();  
 this.loader = null;  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/Toaster.vue***

<template>  
 <Transition name="toast">  
 <section class="toast-container text-center" v-if="visible">  
 <div class="toast d-block m-0">  
 <div  
 class="toast-body alert m-0"  
 :class="status === Status.*ACTIVE* ? 'alert-success' : 'alert-danger'"  
 >  
 {{ this.msg }}  
 </div>  
 </div>  
 </section>  
 </Transition>  
</template>  
  
<script lang="ts">  
import { *Component*, *Prop*, Vue } from "vue-property-decorator";  
import { Status } from "@/../api";  
  
export function *getBanMsg*(status: Status): string {  
 return status == Status.*ACTIVE* ? "Разблокировано" : "Заблокировано";  
}  
  
*// Компонент вывода всплывающих сообщений*@Component({  
 components: {},  
})  
export default class Toaster extends Vue {  
 private Status = Status;  
 @Prop({ default: 1500 }) interval!: number;  
  
 private msg = "";  
 private status: Status = Status.*UNCONFIRMED*;  
 private visible = false;  
 private timerId = 0;  
  
 show(status: Status, msg: string) {  
 this.status = status;  
 this.msg = msg;  
 this.visible = true;  
 *clearTimeout*(this.timerId);  
 this.timerId = *setTimeout*(() => {  
 this.visible = false;  
 }, this.interval);  
 }  
}  
</script>  
  
<style scoped lang="scss">  
.toast-enter-active,  
.toast-leave-active,  
.alert {  
 transition: all 0.3s ease-in-out;  
}  
.toast-enter,  
.toast-leave-to {  
 opacity: 0;  
 transform: translateY(100%);  
}  
</style>

***Файл src/components/UserCard.vue***

<template>  
 <article  
 class="d-flex p-3 flex-column flex-md-row align-items-center align-items-md-stretch"  
 v-if="loader?.data"  
 >  
 <div  
 class="col-6 col-sm-5 col-md-3 p-2 d-flex align-items-center justify-content-center"  
 >  
 <UserThumbnail :loader="loader" class="col-12"></UserThumbnail>  
 </div>  
 <div  
 class="col-12 col-md-9 ps-0 ps-md-3 d-flex flex-column justify-content-between"  
 >  
 <h4 class="row justify-content-between g-0 align-items-start">  
 <span class="col-8">  
 {{ loader.fullName }}  
 </span>  
 <span class="badge bg-secondary rounded-pill col-2">#{{ id }}</span>  
 <font-awesome-icon  
 icon="fa-solid fa-user-gear"  
 class="text-primary col-1 fs-3"  
 role="button"  
 @click="settings"  
 />  
 </h4>  
 <h5 class="text-primary">  
 {{ loader.data.email }}  
 </h5>  
 <h6 class="text-muted">  
 Дата рождения:  
 {{ new Date(loader.data.birthDate).toLocaleDateString() }}  
 </h6>  
 <h6>  
 <span  
 class="badge rounded-pill"  
 :class="['bg-secondary', 'bg-info'][loader.data.role]"  
 >  
 {{ ["Пользователь", "Администратор"][loader.data.role] }}  
 </span>  
 </h6>  
 <h6>  
 <span  
 class="badge rounded-pill"  
 :class="['bg-success', 'bg-warning', 'bg-danger'][loader.data.status]"  
 >  
 {{  
 ["Активный", "Не подтверждённый", "Заблокированный"][  
 loader.data.status  
 ]  
 }}  
 </span>  
 </h6>  
 </div>  
 </article>  
</template>  
  
<script lang="ts">  
import { Component, InjectReactive, Prop, Vue } from "vue-property-decorator";  
import { UserLoader } from "@/loaders";  
import { Views } from "@/router";  
import UserThumbnail from "@/components/UserThumbnail.vue";  
import RedactModal from "@/components/RedactModal.vue";  
  
*// Карточка Пользователя*@Component({  
 components: { UserThumbnail },  
})  
export default class UserCard extends Vue {  
 private Views = Views;  
 @InjectReactive() readonly redact!: RedactModal | null;  
 @Prop({ type: Number, required: true }) readonly id!: number;  
  
 private loader: UserLoader | null = null;  
  
 private async mounted() {  
 this.loader = new UserLoader(this.id);  
 await this.loader.fetch();  
 }  
  
 private settings() {  
 if (this.loader?.data) this.redact?.show(this.loader);  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/components/UserThumbnail.vue***

<template>  
 <router-link  
 :to="{  
 name: Views.USER,  
 params: { user\_id: this.loader.id.toString() },  
 }"  
 class="position-relative card-image"  
 >  
 <font-awesome-icon  
 icon="fa-circle-arrow-left"  
 class="position-absolute text-info open w-100 h-100 bg-white"  
 />  
  
 <font-awesome-icon  
 v-if="!loader.data.photoId"  
 icon="fa-circle-user"  
 class="rounded-circle img-fluid text-primary"  
 />  
 <img v-else :src="loader.avatarUrl" class="rounded-circle img-fluid" />  
 </router-link>  
</template>  
  
<script lang="ts">  
import { Component, Prop, Vue } from "vue-property-decorator";  
import { UserLoader } from "@/loaders";  
import { Views } from "@/router";  
  
*// Иконка пользователя*@Component({})  
export default class UserThumbnail extends Vue {  
 private Views = Views;  
 @Prop({ required: true }) readonly loader!: UserLoader;  
}  
</script>  
  
<style scoped lang="scss">  
.card-image {  
 .open {  
 transition: all 0.3s ease-in-out;  
 top: 0;  
 bottom: 0;  
 left: 0;  
 right: 0;  
 opacity: 0;  
 }  
  
 img,  
 svg {  
 object-fit: cover;  
 aspect-ratio: 1/1;  
 }  
  
 &:hover {  
 .open {  
 opacity: 1;  
 }  
 }  
}  
</style>

***Файл src/views/LoginView.vue***

<template>  
 <form  
 class="login container border shadow rounded-3 col-md-10 mt-5 p-5"  
 @submit.prevent="submit"  
 >  
 <div class="fs-3">  
 <label for="login" class="form-label">Email</label>  
 <input  
 v-model="user.email"  
 type="text"  
 class="form-control"  
 :class="{ 'is-invalid': !valid }"  
 @keydown="reset"  
 id="login"  
 required  
 />  
  
 <label for="pwd" class="mt-3 form-label">Пароль</label>  
 <input  
 v-model="user.password"  
 type="password"  
 class="form-control"  
 :class="{ 'is-invalid': !valid }"  
 @keydown="reset"  
 id="pwd"  
 required  
 />  
 </div>  
 <div class="mt-4 row d-flex justify-content-center">  
 <button type="submit" class="btn btn-primary col-6 fs-3 fw-bold">  
 Вход  
 </button>  
 </div>  
 </form>  
</template>  
  
<script lang="ts">  
import { Component, InjectReactive, Vue } from "vue-property-decorator";  
import axios from "axios";  
import { config, Status, UserAuthData } from "@/../api";  
import Toaster from "@/components/Toaster.vue";  
import { Views } from "@/router";  
  
*// Страница входа*@Component({})  
export default class LoginView extends Vue {  
 @InjectReactive() readonly toaster!: Toaster | null;  
 private user: UserAuthData = { email: "", password: "" };  
 private valid = true;  
  
 private readonly url = new URL(  
 config.endpoints.login,  
 config.server  
 ).toString();  
  
 private async submit() {  
 try {  
 await axios.post<UserAuthData>(this.url, this.user);  
 this.$router.push({ name: Views.HOME });  
 } catch (err) {  
 this.toaster?.show(Status.BLOCKED, "Неверные данные для входа");  
 this.valid = false;  
 }  
 }  
  
 private reset() {  
 this.valid = true;  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/views/UsersListView.vue***

<template>  
 <section class="container mt-3 mb-3">  
 <UsersList :list="users.list"></UsersList>  
 </section>  
</template>  
  
<script lang="ts">  
import { Component, Vue } from "vue-property-decorator";  
import UsersList from "@/components/lists/UsersList.vue";  
import { UsersListLoader } from "@/loaders";  
  
*// Страница списка всех пользователей*@Component({  
 components: { UsersList },  
})  
export default class UsersListView extends Vue {  
 private users: UsersListLoader = new UsersListLoader();  
  
 private mounted() {  
 this.users.fetch();  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/views/UserView.vue***

<template>  
 <section class="container">  
 <UserCard :id="id" class="m-auto" :key="id"></UserCard>  
  
 <nav class="col-lg-8 m-auto mt-2 btn-group w-100">  
 <router-link  
 v-for="link of categories"  
 :key="link[0]"  
 :to="{ name: link[0] }"  
 class="btn btn-outline-primary"  
 exact-active-class="active"  
 >{{ link[1] }}</router-link  
 >  
 </nav>  
  
 <div class="mt-3 mb-3">  
 <router-view v-if="loader?.list" :list="loader.list"></router-view>  
 </div>  
 </section>  
</template>  
  
<script lang="ts">  
import { *Component*, Vue, *Watch* } from "vue-property-decorator";  
import UserCard from "@/components/UserCard.vue";  
import { Route } from "vue-router";  
import { Views } from "@/router";  
import {  
 FriendsLoader,  
 FriendsPostsLoader,  
 ListLoader,  
 PhotosLoader,  
 PostsLoader,  
} from "@/loaders";  
  
*// Страница пользователя*@Component({  
 components: { UserCard },  
})  
export default class UserView extends Vue {  
 private id: number = *parseInt*(this.$route.params.user\_id);  
 private loader: ListLoader | null = null;  
  
 readonly categories = [  
 [Views.*POSTS*, "Новости"],  
 [Views.*PHOTOS*, "Фото"],  
 [Views.*FRIENDS*, "Друзья"],  
 [Views.*FRIENDSPOSTS*, "Новости друзей"],  
 ];  
  
 private mounted() {  
 this.onRouteChange(this.$route);  
 }  
  
 @Watch("$route")  
 onRouteChange(to: Route) {  
 const id = *parseInt*(this.$route.params.user\_id);  
 if (this.id !== id) {  
 this.id = id;  
 } else  
 (this.loader =  
 ((): ListLoader | undefined => {  
 switch (to.name as Views) {  
 case Views.*POSTS*:  
 return new PostsLoader(this.id);  
 case Views.*PHOTOS*:  
 return new PhotosLoader(this.id);  
 case Views.*FRIENDS*:  
 return new FriendsLoader(this.id);  
 case Views.*FRIENDSPOSTS*:  
 return new FriendsPostsLoader(this.id);  
 }  
 })() ?? null)?.fetch();  
 }  
}  
</script>  
  
<style scoped lang="scss"></style>

***Файл src/App.vue***

<template>  
 <div id="app">  
 <header  
 class="header container-fluid pt-3 pb-3 ps-5 pe-5 bg-primary text-light fw-bold fs-2"  
 >  
 <router-link  
 :to="{ name: Views.HOME }"  
 class="text-white text-decoration-none"  
 >  
 <span class="me-2">Администрация</span>  
 <font-awesome-icon icon="fa-wrench" />  
 </router-link>  
 </header>  
 <main class="view col-md-8 col-xl-6 m-auto">  
 <router-view />  
 </main>  
  
 <Toaster class="position-fixed bottom-0 end-0 p-3" ref="toaster" />  
 <RedactModal ref="redact" />  
 </div>  
</template>  
  
<script lang="ts">  
import { Component, ProvideReactive, Vue } from "vue-property-decorator";  
import { Views } from "./router";  
import Toaster from "@/components/Toaster.vue";  
import RedactModal from "@/components/RedactModal.vue";  
  
@Component({  
 components: { RedactModal, Toaster },  
})  
export default class App extends Vue {  
 private Views = Views;  
  
 $refs!: {  
 toaster: Toaster;  
 redact: RedactModal;  
 };  
  
 private mounted() {  
 this.toaster = this.$refs.toaster;  
 this.redact = this.$refs.redact;  
 }  
  
 @ProvideReactive() toaster: Toaster | null = null;  
 @ProvideReactive() redact: RedactModal | null = null;  
}  
</script>  
  
<style lang="scss">  
@import "styles/main";  
</style>

***Файл src/icons.ts***

export {  
 faWrench,  
 faUserGear,  
 faNewspaper,  
 faImages,  
 faUserGroup,  
 faRadio,  
 faCircleUser,  
 faCircleArrowLeft,  
 faBan,  
 faClockRotateLeft,  
} from "@fortawesome/free-solid-svg-icons";

***Файл src/main.ts***

import Vue from "vue";  
import App from "./App.vue";  
import router from "./router";  
import axios from "axios";  
import \* as icons from "./icons";  
import "bootstrap";  
import { library } from "@fortawesome/fontawesome-svg-core";  
import { FontAwesomeIcon } from "@fortawesome/vue-fontawesome";  
  
library.add(icons);  
Vue.component("font-awesome-icon", FontAwesomeIcon);  
  
Vue.config.productionTip = false;  
new Vue({  
 router,  
 render: (h) => h(App),  
}).$mount("#app");  
  
*// Добавить куки ко всем запросам*axios.interceptors.request.use(function (config) {  
 config.withCredentials = true;  
 return config;  
});

***Файл babel.config.js***

module.exports = {  
 presets: ["@vue/cli-plugin-babel/preset"],  
};

***Файл conf.nginx***

server {  
 listen 80;  
 server\_name localhost;  
 return 307 https://localhost$request\_uri;  
}  
  
server {  
 listen 443 http2;  
 server\_name localhost;  
  
 #SSL settings  
 ssl\_certificate /home/vlad/ssl/cert.crt;  
 ssl\_certificate\_key /home/vlad/ssl/key.key;  
  
 charset utf-8;  
 root /home/vlad/WebstormProjects/WebLabs/dist;  
 index index.html;  
  
 location /api {  
 proxy\_pass http://localhost:3000;  
 }  
}