

Министерство науки и высшего образования Российской Федерации  
Муромский институт (филиал)  
Федерального государственного бюджетного образовательного учреждения высшего  
образования  
«Владимирский государственный университет  
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Факультет \_\_\_\_\_ ИТР \_\_\_\_\_

Кафедра \_\_\_\_\_ ПИН \_\_\_\_\_

## КУРСОВАЯ РАБОТА

По Разработке корпоративных приложений

Тема Веб-приложение «Социальная сеть»

Руководитель

Кульков Я.Ю.  
(фамилия, инициалы)

\_\_\_\_\_  
(подпись) (дата)

Студент ПИН - 121  
(группа)

Ермилов М.В.  
(фамилия, инициалы)

\_\_\_\_\_  
(подпись) (дата)

Муром 2024



В современном мире существует множество технологий для создания сложных систем. В данном проекте рассмотрена разработка социальной сети с использованием фреймворка Spring, включая реализацию аутентификации, управления сессиями, отправки уведомлений и поддержки веб-сокетов для обмена сообщениями.

In today's world, there are many technologies for building complex systems. This project explores the development of a social network using the Spring framework, including the implementation of authentication, session management, notification sending, and support for WebSockets for message exchange.

## Содержание

Введение.....	6
1 Анализ технического задания.....	7
2 Разработка моделей данных.....	9
3 Проектирование работы системы.....	12
4 Разработка и реализация системы.....	14
5 Тестирование системы.....	18
Заключение.....	19
Список используемой литературы.....	20
Приложение 1. Модели данных.....	21
Приложение 2. Основные ссылки.....	22
Приложение 3. Скриншоты программы.....	23

					МИВлГУ 09.03.04 - 0.012			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Ермилов М.В.			«Социальная сеть»	Лит.	Лист	Листов
Провер.		Кульков Я.Ю.					5	29
Реценз.						МИ ВлГУ ПИН-121		
Н. Контр.								
Утверд.								

## Введение

В современном цифровом мире социальные сети стали неотъемлемой частью жизни, предоставляя пользователям возможности для обмена сообщениями, публикации контента, участия в сообществах и поддержания социальных связей. Успех социальных платформ во многом зависит от удобства интерфейса и надежной работы серверной части, которая обеспечивает обработку и хранение данных, защищает личную информацию и обеспечивает связь между пользователями в режиме реального времени.

Одним из ключевых компонентов любой социальной сети является серверное приложение, предоставляющее интерфейсы (API) для доступа к данным и функциональности, связанной с пользователями, их профилями, публикациями, комментариями и другими элементами взаимодействия. API, как посредник между клиентскими приложениями и сервером, позволяет организовать обмен данными в стандартизированном формате, таком как JSON, обеспечивая независимость клиентских приложений от серверной реализации.

Целью данного курсового проекта является разработка серверного веб-приложения "Социальная сеть" на базе Java и фреймворка Spring. Данное приложение должно реализовывать основные функции социальной сети, включая регистрацию и аутентификацию пользователей, публикацию контента, комментарии, отправку сообщений, а также возможность взаимодействия с друзьями. Фреймворк Spring выбран для этой задачи благодаря его гибкости и наличию встроенных инструментов для разработки API, обеспечения безопасности и удобного управления зависимостями.

Для достижения поставленной цели проект будет включать несколько этапов, в том числе анализ требований, проектирование структуры данных, разработку и тестирование основных функциональных модулей. Важно также обеспечить безопасность передаваемых данных, реализовать механизм управления сессиями и оптимизировать производительность системы.

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

## 1. Анализ технического задания

Разработка серверной части социальной сети требует создания надежного API и инфраструктуры для эффективного обмена данными между клиентом и сервером. Основные функциональные и технические аспекты включают авторизацию, управление сессиями, обработку сообщений, уведомлений, а также взаимодействие с базой данных.

### Основные требования:

- **Архитектура клиент-сервер и формат данных:**
  - Обмен данными между клиентом и сервером осуществляется через HTTP POST-запросы с использованием JSON-формата для стандартизации структуры передаваемых данных.
  - Приложение использует преимущественно клиентский рендеринг с серверной передачей стартовых данных в тегах `<script>`, что позволяет ускорить загрузку страниц.
  - Динамическая генерация страниц, включая основную, происходит с помощью шаблонизатора Thymeleaf, а обновления интерфейса обрабатываются на клиенте с использованием JavaScript.
- **Управление сессиями и авторизация:**
  - Сессии управляются через механизмы куки, что упрощает процесс авторизации и позволяет поддерживать контекст пользователя между запросами.
  - Куки обеспечивают автоматическую идентификацию пользователя при каждом запросе и позволяют работать с сессиями без повторной авторизации.
- **Передача сообщений и уведомлений:**
  - Система использует WebSocket для реализации обмена сообщениями и уведомлениями в реальном времени, что сокращает

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

задержки при передаче данных и улучшает взаимодействие с пользователями.

- **База данных, кэширование и взаимодействие с ней:**

- Данные пользователей, сообщений и фотографий хранятся в базе данных MariaDB, а информация о пользователях и логинах кэшируется для повышения производительности.
- Система использует представления MariaDB для выборки данных, минимизируя количество запросов и повышая скорость работы приложения.

- **Модели данных и оптимизация запросов:**

- Модели данных, такие как Photo и User, спроектированы для эффективной работы с вложенными данными и минимизации количества запросов к базе данных.
- Данные о пользователях передаются вместе с вложенной информацией, например, аватаром и обложкой, что снижает количество запросов к базе и повышает скорость обработки.



## 2. Разработка моделей данных

Для реализации приложения была спроектирована **схема базы данных** (Приложение 1), на основе которой были созданы **РОЖО-классы** (Plain Old Java Object). Эти классы представляют собой модели данных, которые обеспечивают корректное отображение информации пользователям и удобное взаимодействие с бизнес-логикой приложения.

Ниже приведено описание ключевых моделей:

### 1. Login

Модель **Login** отвечает за хранение учетных данных пользователей:

```
public class Login {  
    private int id;          // Уникальный идентификатор пользователя  
    private String email;    // Электронная почта пользователя  
    private String password; // Пароль пользователя  
}
```

### 2. Message

Модель **Message** расширяет функциональность **Response** и используется для представления сообщений в системе:

```
public class Message extends Response {  
    private User sender;      // Объект отправителя сообщения  
    private int senderId;     // Идентификатор отправителя  
    private int id;           // Уникальный идентификатор сообщения  
    private List<PhotoSimple> photo; // Список вложенных изображений  
    private long date;        // Дата создания сообщения  
    private long date_edit;   // Дата последнего редактирования  
    private String text;      // Текст сообщения  
}
```

### 3. PhotoSimple

Базовый класс для представления фотографии:

```
public class PhotoSimple {  
    protected String file; // Путь к файлу изображения  
}
```

### 4. Photo

Модель **Photo** наследуется от **PhotoSimple** и добавляет дополнительные свойства:

```
public class Photo extends PhotoSimple {  
    private int album; // Идентификатор альбома, к которому относится фото  
    private int id;    // Уникальный идентификатор фотографии  
}
```

### 5. Post

Модель **Post** наследуется от **Rating** и используется для отображения публикаций пользователей:

```
public class Post extends Rating {  
    private User user;          // Объект пользователя, создавшего пост  
    private int userId;         // Идентификатор пользователя  
    private int id;             // Уникальный идентификатор поста  
    private String text;        // Текст публикации  
    private List<Photo> photo;   // Список прикрепленных фотографий  
    private long date;          // Дата создания публикации  
    private long date_edit;     // Дата последнего редактирования  
}
```

## 6. User

Модель **User** описывает основную информацию о пользователе системы:

```
public class User {  
    private int online = -1;        // Статус "онлайн" пользователя (-1 по  
    // умолчанию)  
    private int id;                // Уникальный идентификатор пользователя  
    private String name;           // Имя пользователя  
    private String status;         // Статус пользователя  
    private LocalDate dateBirth;   // Дата рождения  
    private String tag;            // Уникальный тег пользователя  
    private int countFriend;       // Количество друзей  
    private int countSubscribers;  // Количество подписчиков  
    private int countSubscriptions; // Количество подписок  
    private Photo icon;            // Аватар пользователя  
    private Photo cover;           // Обложка профиля  
    private boolean isVerified;    // Статус верификации  
    private boolean isGroup;       // Является ли это группой  
    private boolean isPopular;     // Статус популярности  
    private boolean isMusician;    // Статус музыканта  
}
```

### Заключение

Созданные POJO-классы обеспечивают структурированное представление данных, необходимых для работы системы, и позволяют взаимодействовать с базой данных через ORM-инструменты, такие как **Hibernate** или **Spring Data JPA**. Это решение повышает читаемость кода, улучшает поддержку проекта и способствует гибкой расширяемости функциональности.

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

### 3. Проектирование работы системы

#### 1. Аутентификация и управление сессиями:

- **Цель:** Обеспечить безопасность работы пользователей путем аутентификации и управления сессиями.
- **Механизм:**
  - Использование JWT-токенов и сессий для аутентификации пользователей.
  - Хранение активных сессий на сервере в памяти и на диске для долгосрочного хранения.
  - Автоматическое удаление сессий с истекшим сроком действия.

#### Реализация:

Код **JwtAuthenticationFilter** обрабатывает сессии, сохраняет их на диск и обеспечивает доступ через cookies:

@PostConstruct

```
void init() { ... } // Инициализация сессий
```

```
public void login(HttpSession session, HttpServletResponse response, int id, boolean fast) { ... }
```

```
public void logout(HttpSession session, HttpServletResponse response, HttpServletRequest request) { ... }
```

- **Сценарий использования:** При успешной аутентификации пользователь получает sessionId, сохраняемый в cookie и на диске.

#### 2. Обработка Email-уведомлений:

- **Цель:** Информировать пользователей о критически важных действиях (вход, регистрация).
- **Механизм:**
  - Использование **EmailService** для отправки писем с шаблонами.

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

- Уведомления включают информацию об устройстве, браузере и IP-адресе пользователя.

#### Реализация:

```
public void sendLoginConfirm(HttpServletRequest request, String email, String name, String code) { ... }
```

```
public void sendRegistration(String email, String name, String code) { ... }
```

- **Сценарий использования:** При попытке входа или регистрации пользователю отправляется код подтверждения или уведомление о входе.

### 3. Обработка WebSocket сообщений:

- **Цель:** Поддержка реального времени для уведомлений и статуса активности пользователей.
- **Механизм:**
  - Обработка WebSocket-соединений с проверкой сессий для аутентификации.
  - Управление статусом "онлайн" и отправка уведомлений пользователям.

#### Реализация:

```
public void afterConnectionEstablished(WebSocketSession session) { ... }
```

```
public void sendNotification(int userId, String message) { ... }
```

- **Сценарий использования:** Сервер отправляет сообщение пользователю при изменении статуса или появлении новых событий.

## 4. Разработка системы

### 1. Реализация аутентификации и сессий:

- **Ключевые компоненты:**

- **JwtAuthenticationFilter:** Управление логином, выходом и проверкой сессий.
- **Cookie:** Использование `sessionId` для идентификации пользователя.
- **Сохранение сессий на диск:** ID пользователя и срок действия сохраняются для долгосрочной работы.

**Код:**

```
public void login(HttpSession session, HttpServletResponse response, int id, boolean
fast) {
    session.setMaxInactiveInterval(fast ? 0 : 60 * 60 * 24 * 30);
    String sessionId = session.getId();
    if (!auth.containsKey(sessionId)) {
        Authentication authentication = new UsernamePasswordAuthenticationToken(id,
null, List.of(new SimpleGrantedAuthority("user")));
        auth.put(sessionId, authentication);
        if (!fast) {
            try (DataOutputStream dos = new DataOutputStream(new
FileOutputStream(DIR + sessionId))) {
                dos.writeInt(id);
                long expiryTime = System.currentTimeMillis() + (long)
session.getMaxInactiveInterval() * 1000;
                dos.writeLong(expiryTime);
            } catch (IOException e) { }
        }
    }
}
```

## 2. Разработка Email-уведомлений:

- **EmailService** используется для отправки писем с шаблонизацией.
- Шаблоны включают данные о пользователе и предупреждения о безопасности.

### Код:

```
public void sendEmailWithTemplate(String to, String subject, String fragment,
Map<String, Object> variables) throws MessagingException {
    variables.put("signature", "С уважением, администрация ADAPT-KEY.");
    Context context = new Context();
    context.setVariables(variables);
    context.setVariable("contentFragment", fragment);

    String htmlContent = templateEngine.process("email/pattern", context);
    MimeMessage message = emailSender.createMimeMessage();
    MimeMessageHelper helper = new MimeMessageHelper(message, true, "UTF-8");

    helper.setFrom(new InternetAddress("confirm@adapt-key.com", "ADAPT-KEY"));
    helper.setTo(to);
    helper.setSubject(subject);
    helper.setText(htmlContent, true);
    emailSender.send(message);
}
```

### 3. Разработка WebSocket-сообщений:

- Управление соединениями:
  - Аутентификация через cookies.
  - Отправка сообщений пользователям с использованием WebSocket.

**Код:**

```
@Override
public void afterConnectionEstablished(WebSocketSession session) throws
IOException {
    Map<String, String> cookiesMap =
parseCookies(session.getHandshakeHeaders().get(HttpHeaders.COOKIE));
    String sessionId1 = cookiesMap.getDefault("JSESSIONID", null);
    String sessionId2 = cookiesMap.getDefault("sessionId", null);

    int userId = jwt.getId(sessionId1, sessionId2);
    if (userId <= 0) {
        session.close(CloseStatus.NOT_ACCEPTABLE);
        return;
    }
    socketCache.put(session, userId);
    socket.computeIfAbsent(userId, k -> Collections.synchronizedList(new
ArrayList<>())).add(session);
}
```

- **Рассылка уведомлений:**

```
public CompletableFuture<Void> send(int userId, String message) {
    return CompletableFuture.runAsync(() -> {
        if (socket.containsKey(userId)) {
            for (WebSocketSession session : socket.get(userId)) {
                try {
                    session.sendMessage(new TextMessage(message));
                }
            }
        }
    });
}
```

					МИВЛГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16



```
        } catch (IOException e) { }  
    }  
}  
});  
}
```

					МИВлГУ 09.03.04 – 0.012	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

## 5. Тестирование

Тестирование системы осуществлялось на нескольких уровнях, обеспечивая стабильность работы и безопасность всех компонентов:

### 1. Тестирование API

Для проверки API сервисов использовались специализированные утилиты, позволяющие отправлять HTTP-запросы и анализировать полученные ответы в формате JSON. Это позволяло отлаживать и верифицировать корректность работы серверной логики и передачи данных.

### 2. Тестирование веб-интерфейса

Веб-интерфейс подвергался как полному, так и частичному тестированию. Особое внимание уделялось функциональности основных пользовательских сценариев, а также валидации и обработке данных.

### 3. Кросс-браузерное тестирование

Система разрабатывалась с учётом конечных пользователей, поэтому тесты проводились в различных браузерах (Chrome, Firefox, Opera и других). При этом использовались встроенные инструменты разработчика для анализа сетевых запросов, обработки JSON-ответов сервера и устранения ошибок на клиентской стороне.

### 4. Тестирование взаимодействия с живыми пользователями

После реализации основной части системы тестирование проводилось в приватной локальной сети с привлечением реальных пользователей. Это позволило получить обратную связь о работе системы, выявить ошибки взаимодействия и протестировать безопасность приложения при реальной нагрузке.

Таким образом, многоуровневый подход к тестированию обеспечил высокую надёжность и производительность системы, а также соответствие требованиям пользователей и безопасности данных.

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

## Заключение

В ходе работы было разработано серверное ядро для социальной сети, которое объединяет в себе широкий спектр функциональных возможностей. Основой системы стала надежная архитектура, реализованная с использованием Spring Framework, что позволило обеспечить безопасность, масштабируемость и высокую производительность.

Особое внимание уделено управлению пользовательскими сессиями с сохранением их состояния на диске, что повышает надежность и удобство работы. Добавлены инструменты аутентификации и авторизации, а также интеграция с email-сервисами для подтверждения регистрации, восстановления доступа и уведомлений.

Для улучшения пользовательского опыта и взаимодействия была реализована поддержка WebSocket, что позволяет оперативно доставлять уведомления в реальном времени. Завершающим этапом стало тестирование, включающее использование автоматизированных инструментов, браузерных средств отладки и проверку системы в реальных условиях с привлечением живых пользователей.

Результатом стала гибкая, устойчивая к нагрузкам платформа, готовая к дальнейшему развитию и внедрению новых возможностей.

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

## Список литературы

Блох, Дж. Java. Эффективное программирование / Дж. Блох ; перевод В. Стрельцов ; под редакцией Р. Усманов. — 2-е изд. — Саратов : Профобразование, 2019. — 310 с. — ISBN 978-5-4488-0127-3. — Текст : электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: <http://www.iprbookshop.ru/89870.html>

Свистунов, А. Н. Построение распределенных систем на Java : учебное пособие / А. Н. Свистунов. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 316 с. — ISBN 978-5-4497-0940-0. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/102045.html>

Мухамедзянов, Р. Р. JAVA. Серверные приложения / Р. Р. Мухамедзянов. — Москва : СОЛОН-Р, 2016. — 336 с. — ISBN 5-93455-134-5. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/90352.html>

Кириченко, А. В. Динамические сайты на HTML, CSS, Javascript И Bootstrap. Практика, практика и только практика / А. В. Кириченко, Е. В. Дубовик. — Санкт-Петербург : Наука и Техника, 2018. — 272 с. — ISBN 978-5-94387-763-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: <https://www.iprbookshop.ru/77578.html>

## Приложение 1 — Модели данных

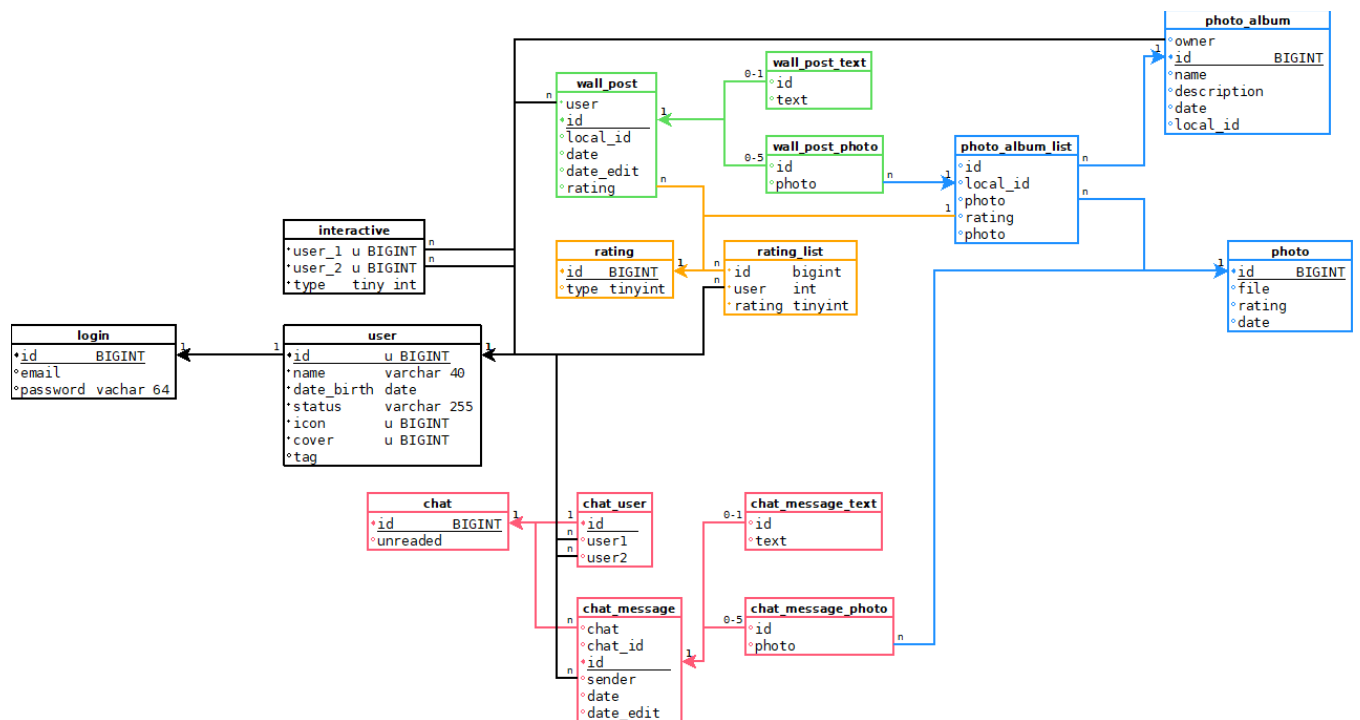


Рисунок 1 — Модель данных

## Приложение 2 — Основные ссылки

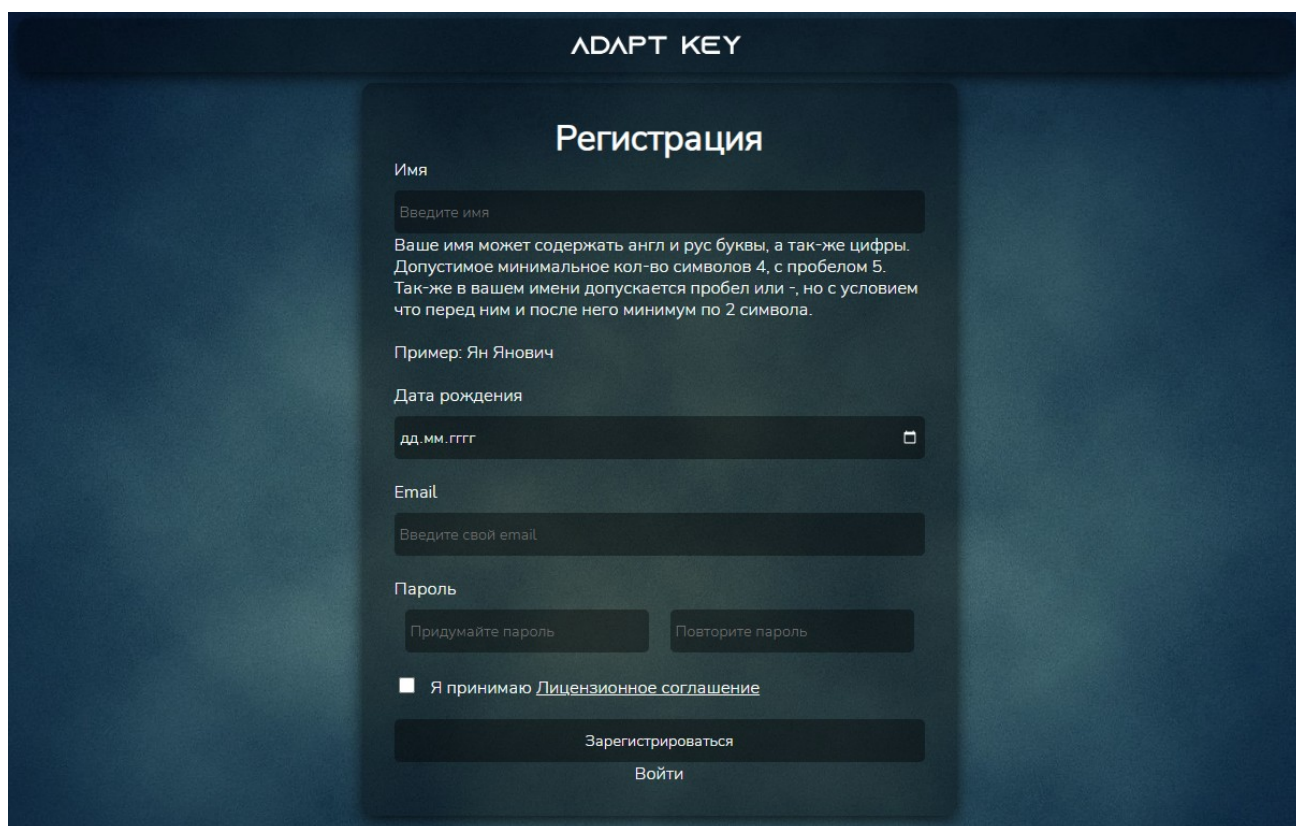
Java Spring код [Электронный ресурс]: <https://github.com/m9studio/AdaptKey-java> (дата обращения: 21 декабря 2024).

Web код [Электронный ресурс]: <https://github.com/m9studio/AdaptKey-web> (дата обращения: 21 декабря 2024).

Основной сайт [Электронный ресурс]: <https://adapt-key.com> (дата обращения: 21 декабря 2024).

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

## Приложение 3 — Скриншоты программы



ADAPT KEY

### Регистрация

Имя

Введите имя

Ваше имя может содержать англ и рус буквы, а так-же цифры. Допустимое минимальное кол-во символов 4, с пробелом 5. Так-же в вашем имени допускается пробел или -, но с условием что перед ним и после него минимум по 2 символа.

Пример: Ян Янович

Дата рождения

дд.мм.гггг

Email

Введите свой email

Пароль

Придумайте пароль

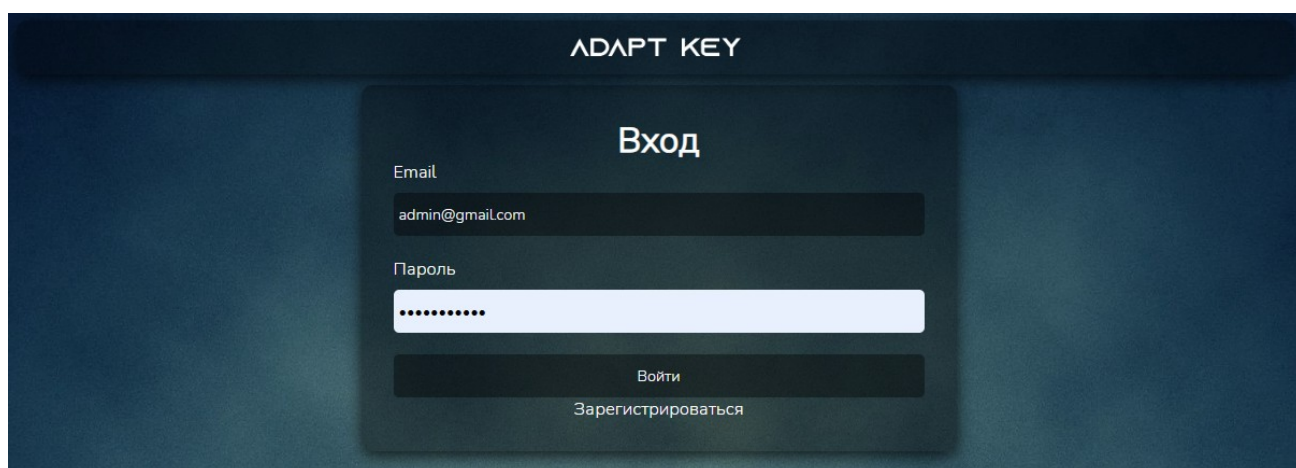
Повторите пароль

☐ Я принимаю [Лицензионное соглашение](#)

Зарегистрироваться

Войти

Рисунок 2 — Страница регистрации



ADAPT KEY

### Вход

Email

admin@gmail.com

Пароль

.....

Войти

Зарегистрироваться

Рисунок 3 — Страница входа

					МИВлГУ 09.03.04 – 0.012	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

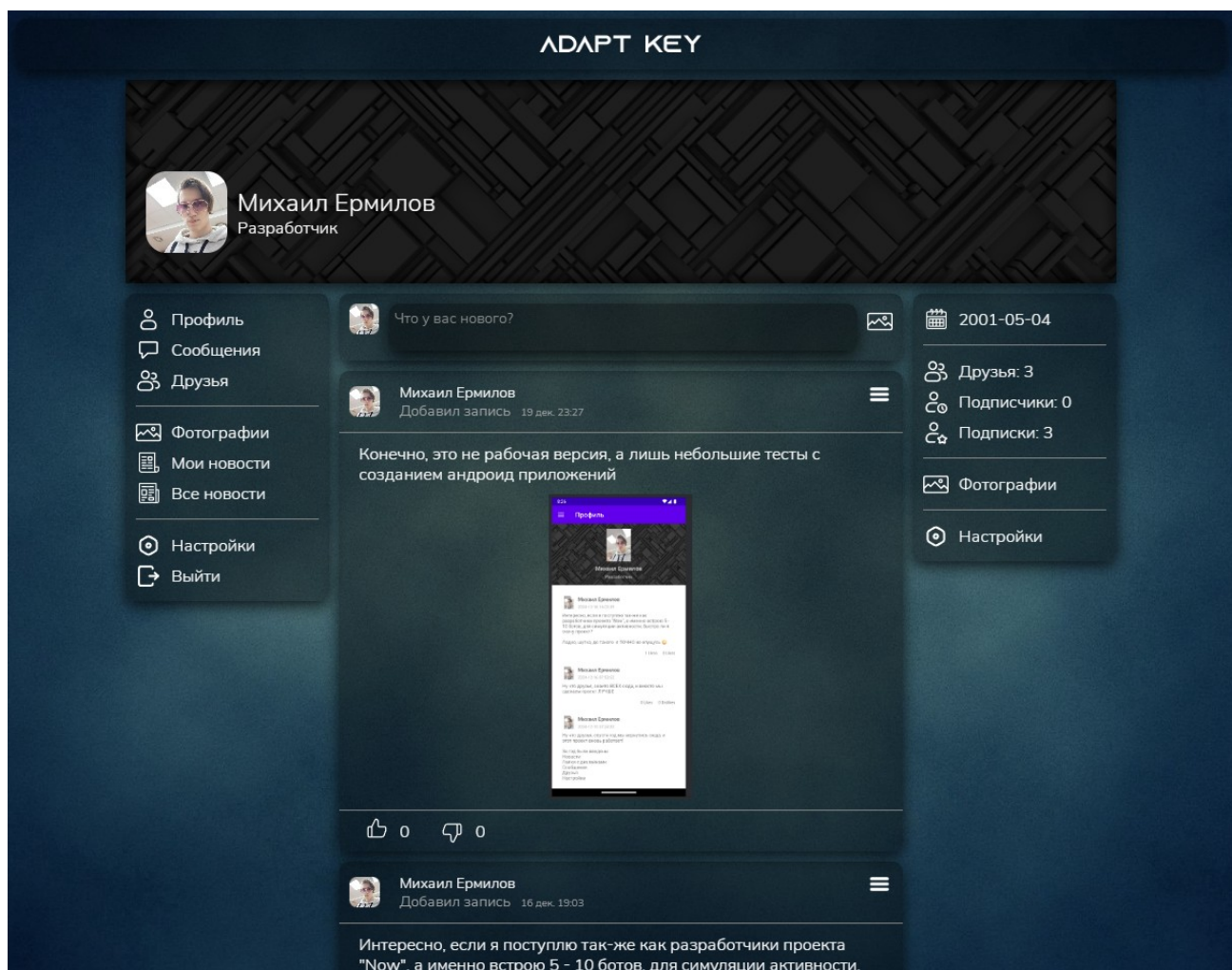


Рисунок 4 — Страница пользователя



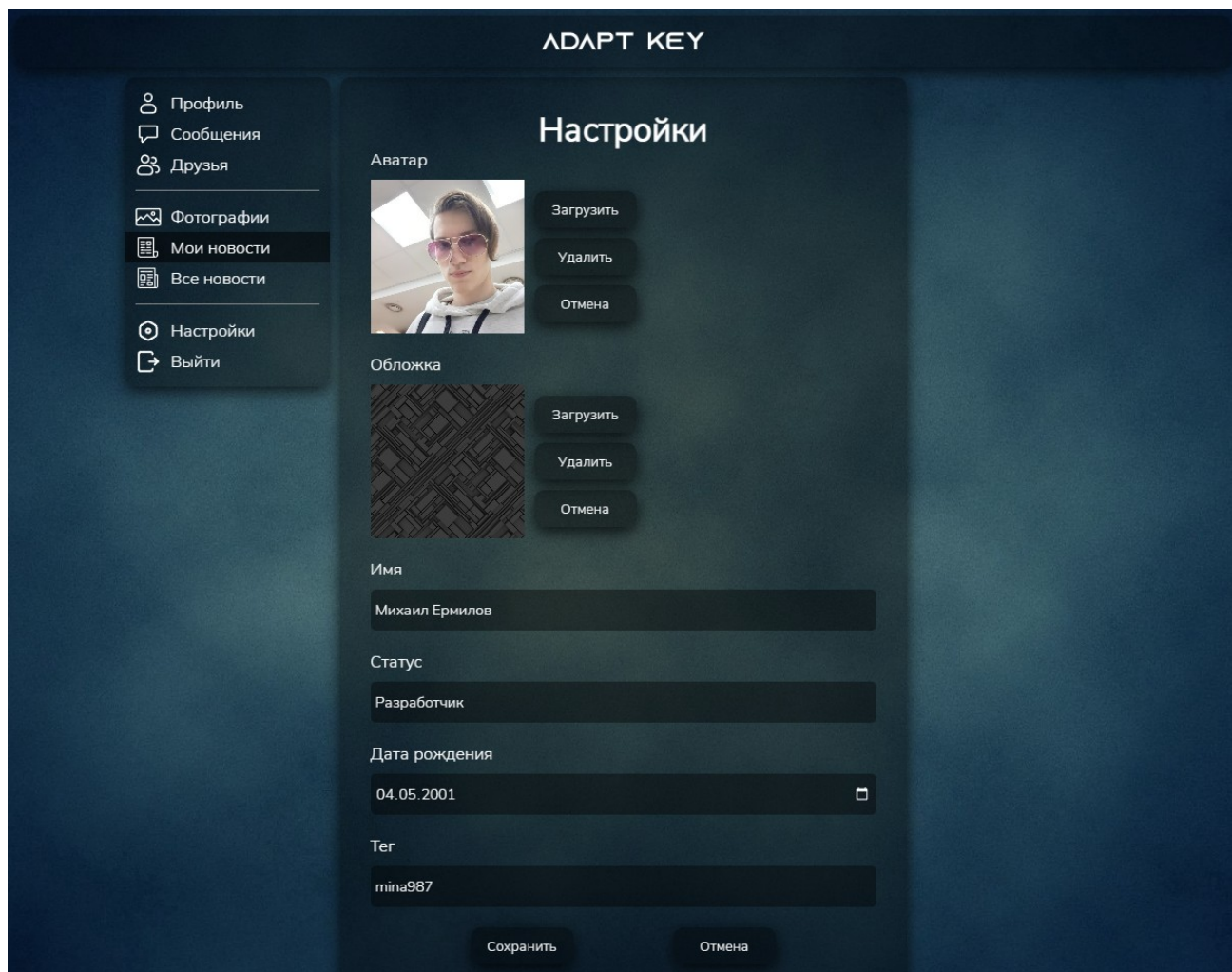


Рисунок 5 — Настройки пользователя

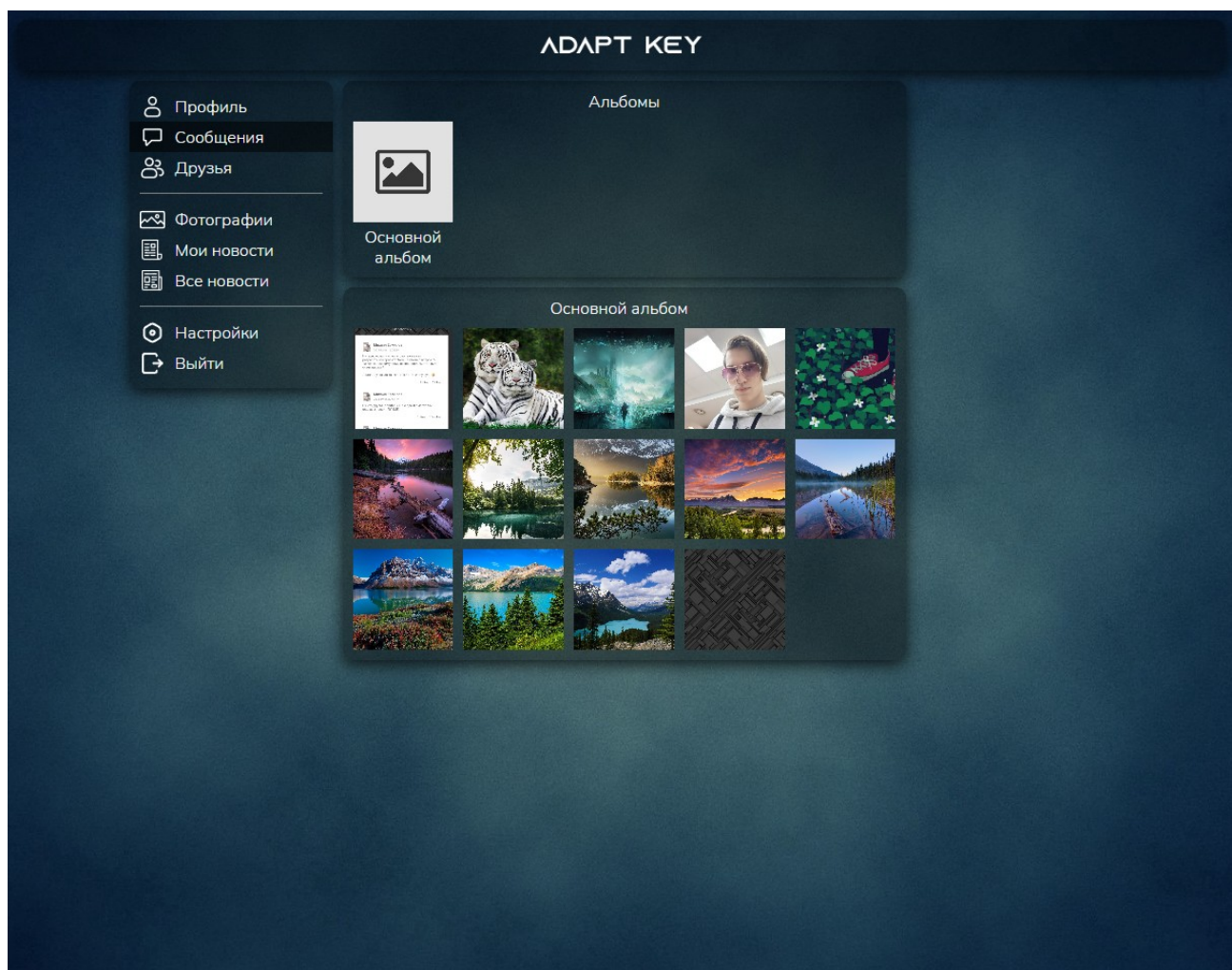


Рисунок 6 — Фотографии пользователя

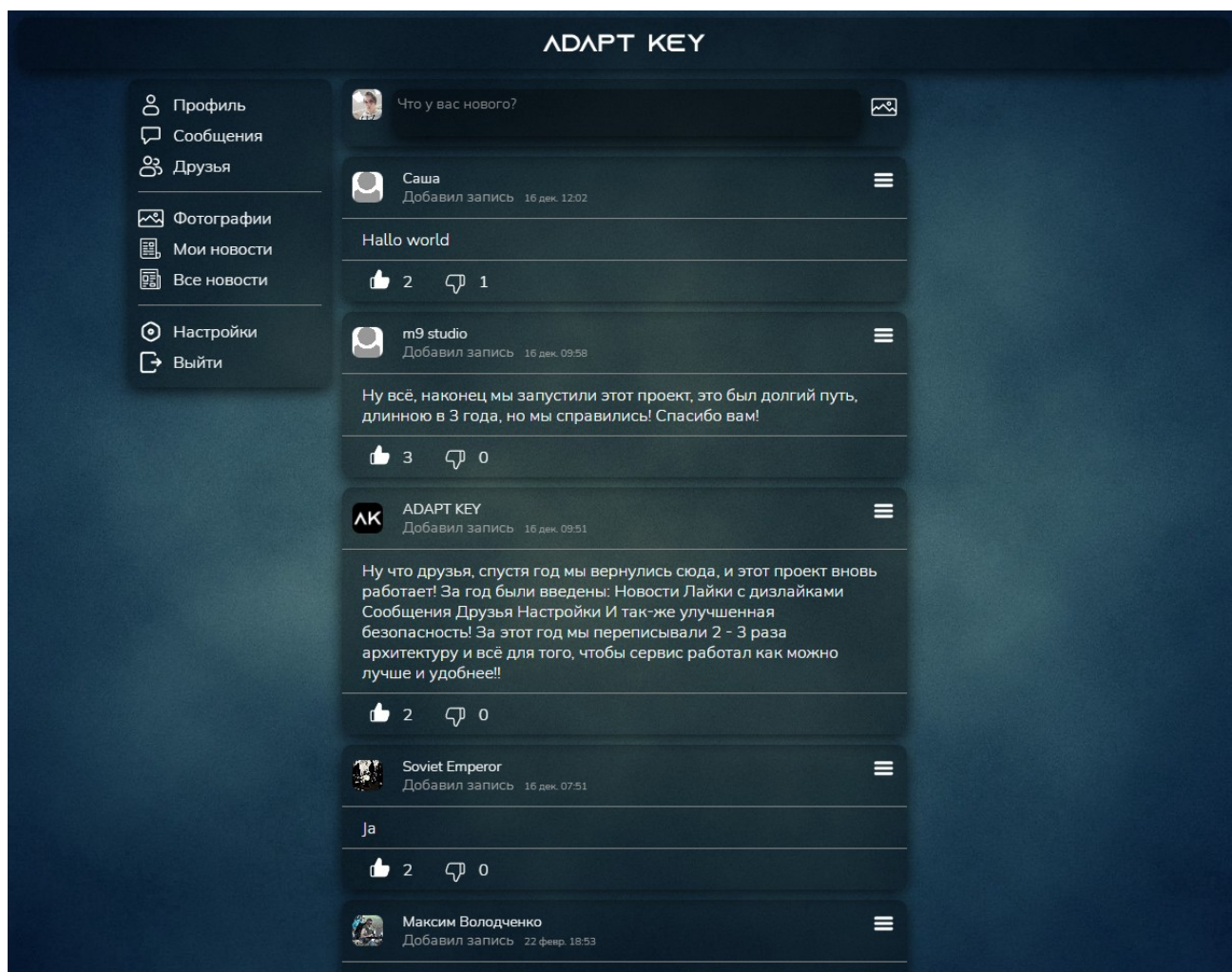


Рисунок 7 — Лента новостей



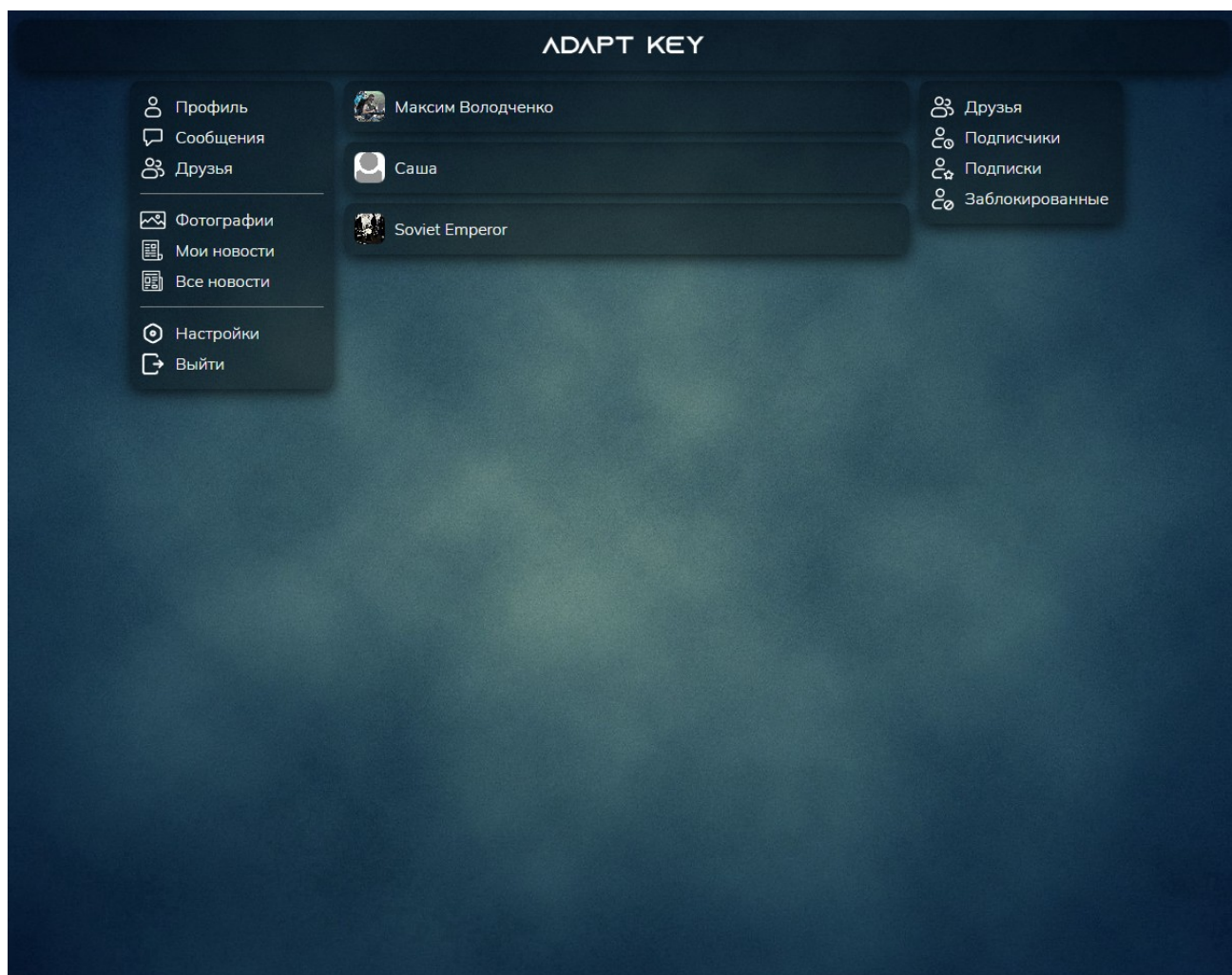


Рисунок 8 — Друзья пользователя

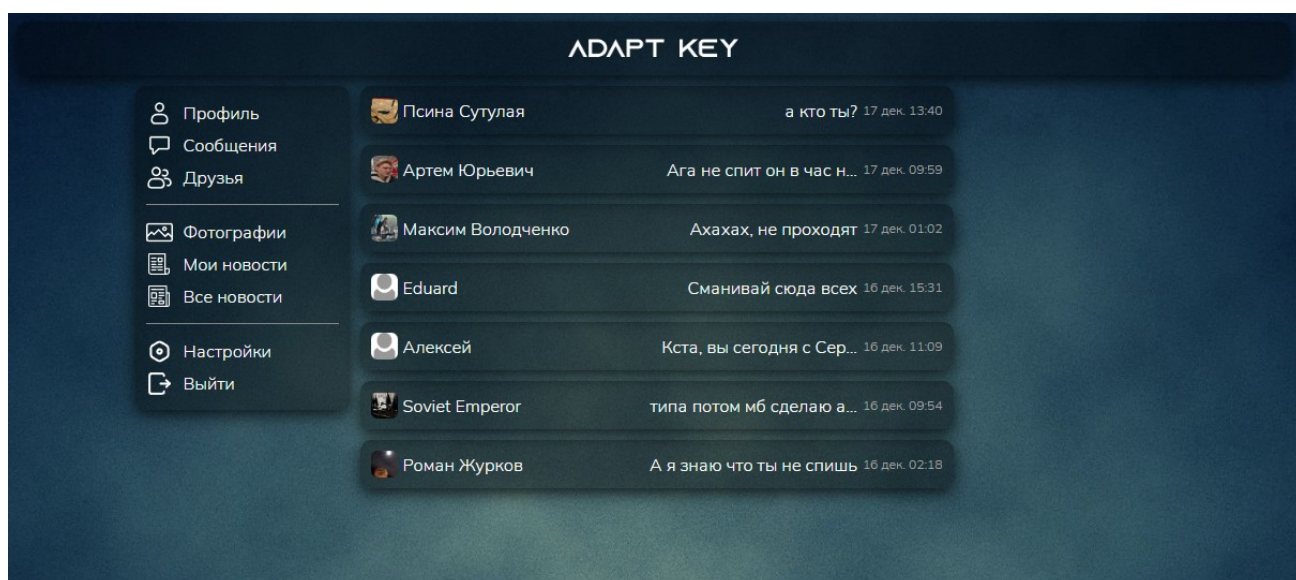


Рисунок 9 — Чаты пользователя

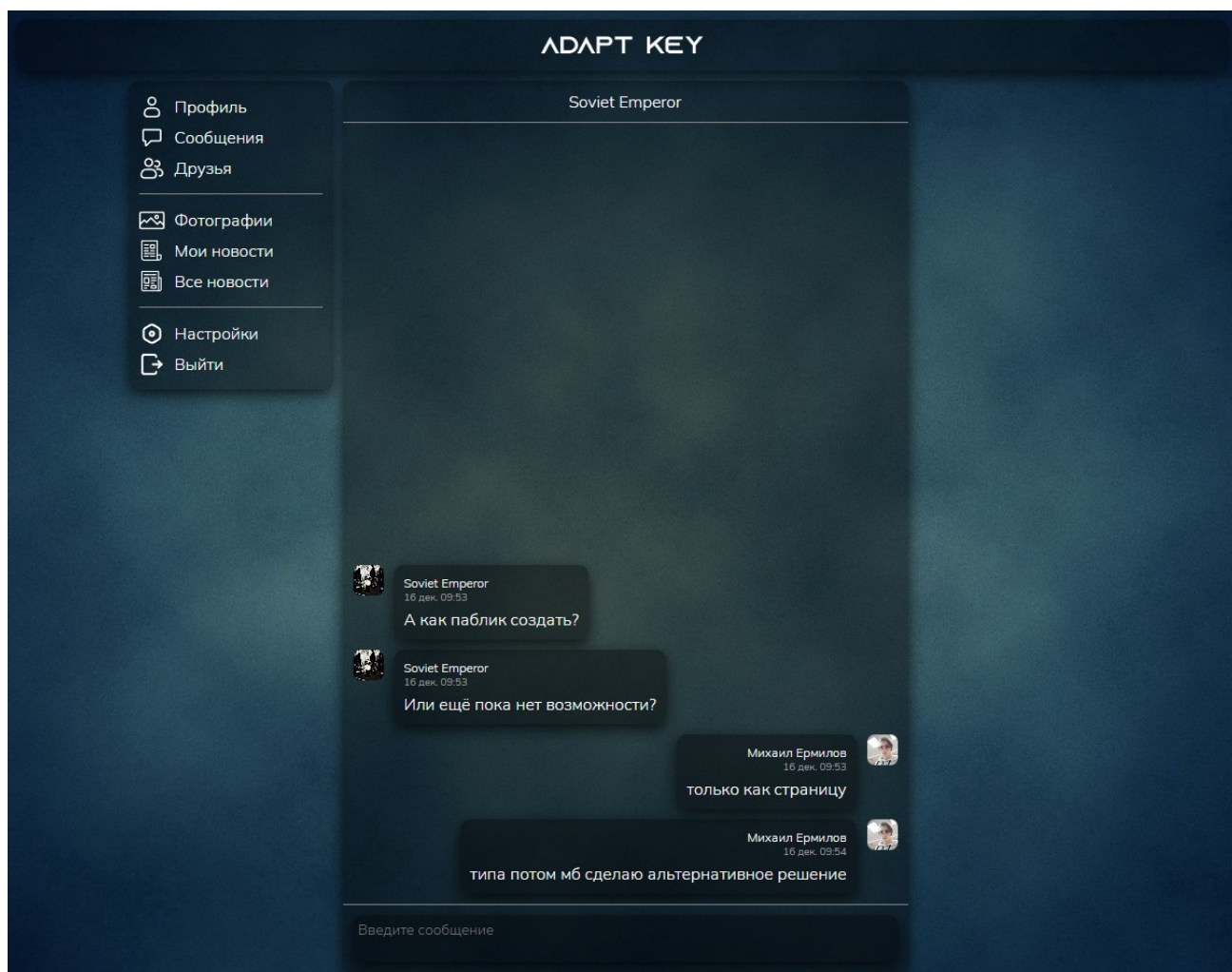


Рисунок 10 — Чат с пользователем