МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ШКОЛА КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ПРОГРАМММНОГО ОБЕСПЕЧЕНИЯ

ОТЧЕТ

О РАЗРАБОТКЕ ПРОЕКТА ДЛЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ ПО ДИСЦИПЛИНЕ

«ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ»

на тему “Разработка сервиса формирования списков наиболее подходящих пользователей на основе близости социальных цифровых портретов”

|  |  |  |
| --- | --- | --- |
| Выполнил  обучающийся 4 курса,  МОиАИС-\_\_\_\_\_\_ группы | (подпись) | Максимюк Александр  Евгеньевич |
| Выполнил  обучающийся 4 курса,  МОиАИС-\_\_\_\_\_\_ группы | (подпись) | Абдиреуков Самат  Талгатулы |
| Научный руководитель | (подпись) | Павлова Елена  Александровна |

|  |  |  |
| --- | --- | --- |
|  |  |  |

Защищен: «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 202\_\_ г.

Результаты защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка)  (подпись руководителя проекта)

# ОГЛАВЛЕНИЕ

[Глава 1. 2](#_Toc185209061)

[1.1 Описание специфики предметной области . 2](#_Toc185209062)

[1.2 Актуальность проекта. 4](#_Toc185209063)

[Глава 2. 5](#_Toc185209064)

[2.1 Идея проекта. 5](#_Toc185209065)

[2.2 Требования к проекту и учитываемые ограничения. 6](#_Toc185209066)

[Глава 3. 8](#_Toc185209069)

[3.1 Теоретическое обоснование проекта. 8](#_Toc185209070)

[3.2 Практическое обоснование проекта. 11](#_Toc185209072)

[3.3 Собственные практические решения. 12](#_Toc185209073)

[3.4 Особенности разработки. 18](#_Toc185209074)

[3.5 Функциональные возможности проекта. 19](#_Toc185209075)

[3.6 Пользовательский интерфейс. 20](#_Toc185209076)

[Список используемой литературы 24](#_Toc185209077)

# Глава 1.

## 1.1 Описание специфики предметной области .

Специфика предметной области проекта связана с анализом и обработкой данных пользователей социальных сетей, в частности, ВКонтакте, с целью улучшения поиска друзей и формирования рекомендаций на основе схожести социальных цифровых портретов. ВКонтакте предоставляет широкие возможности для получения информации о пользователях, включая их демографические данные (возраст, пол, местоположение), интересы, увлечения и активность в группах и других сервисах. Однако алгоритмы поиска в социальных сетях (в т.ч. VK) ограничиваются только базовыми параметрами, такими как пол и возраст, что снижает точность подбора пользователей с похожими интересами.

Основной задачей проекта является разработка алгоритмов, которые смогут учитывать не только стандартные параметры, но и интересы и увлечения пользователей, что позволит предложить более точные и персонализированные рекомендации. Для этого предполагается использование методов анализа данных, таких как алгоритмы схожести (например, косинусное расстояние или методы машинного обучения), которые будут вычислять степень сходства между пользователями по интересам. Такие подходы могут значительно улучшить процесс поиска друзей и создания социальных связей, делая его более удобным и эффективным.

Важным аспектом является использование данных из социальной сети ВКонтакте через VK API, которое предоставляет доступ к информации о пользователях и их активности. Однако из-за ограничений в данных, которые могут быть неполными или не всегда точными, для улучшения качества рекомендаций планируется внедрение алгоритмов предсказания недостающих параметров, таких как пол или возраст. В процессе реализации проекта также необходимо учитывать вопросы конфиденциальности и защиты данных, поскольку использование персональной информации требует строгого соблюдения стандартов безопасности и прозрачности в работе с данными пользователей.

## 1.2 Актуальность проекта.

Существующие алгоритмы поиска пользователей в социальных сетях не предоставляют инструментов, с помощью которых можно учесть интересы человека при составлении выборки. Это усложняет процесс поиска новых знакомств студентов, для которых данная тема актуальна.

Потенциальными пользователями сервиса являются студенты, которым важно находить новых друзей для совместного участия в проектах, учебе, мероприятиях или просто для общения. Также целевой аудиторией выступают пользователи с узкими или редкими интересами, для которых поиск единомышленников стандартными методами оказывается практически невозможным.

Система должна будет учитывать увлечения и интересы пользователей, предлагать действительно подходящих людей и при этом оставаться удобной, быстрой и конфиденциальной. Важно, чтобы поиск был интуитивно понятным, с возможностью гибкой настройки параметров и высокой точностью рекомендаций.

# Глава 2.

## 2.1 Идея проекта.

Разработать сервис на базе социальной сети ВКонтакте, для нахождения друзей, позволяющий осуществлять поиск на основе схожести социальных цифровых портретов, которые включают в себя интересы и увлечения пользователей, а также обычные параметры: пол, возраст и др. с возможностью вычисления значений для них. Основная цель состоит в том, чтобы сделать процесс поиска новых знакомств более точным, удобным и персонализированным.

Для достижения этой цели предполагается использовать данные, предоставляемые социальной сетью ВКонтакте, а также разработать алгоритмы анализа и сравнения социальных цифровых портретов. Благодаря интеграции с VK API система сможет извлекать актуальную информацию о пользователях, а специализированные алгоритмы будут обрабатывать данные для определения степени их схожести. В случае отсутствия некоторых параметров, таких как возраст или пол, планируется внедрение алгоритмов предсказания, которые позволят дополнить недостающую информацию на основе других данных пользователя.

Сервис предложит удобный веб-интерфейс, в котором пользователи смогут задавать параметры поиска, такие как диапазон возраста, интересы и прочие критерии. Это обеспечит гибкость и возможность адаптации под индивидуальные потребности. Таким образом, проект решает проблему отсутствия инструментов для учета интересов при поиске знакомств, создавая решение, которое соответствует требованиям и ожиданиям пользователейзначально наша схема базы данных выглядела следующим образом.

## 2.2 Требования к проекту и учитываемые ограничения.

По данному проекту определены следующие требования :

* Поиск по интересам и увлечениям: Система должна учитывать не только демографические данные (пол, возраст, город и т.д.), но и интересы пользователей, например, хобби, любимые темы, занятия.
* Анализ социальных цифровых портретов: Необходим алгоритм для извлечения и анализа информации о пользователе (интересы, увлечения, предпочтения).
* Рекомендации на основе схожести: Важным является алгоритм, который будет искать пользователей с максимально схожими социальными цифровыми портретами, что улучшит точность рекомендаций.
* Интерфейс для пользователя: Простота и удобство поиска, возможность фильтровать результаты по интересам, полю, возрасту и другим параметрам.

### 2 . Ограничения:

* Доступ к данным: Необходимо учитывать, что доступ к данным пользователей может быть ограничен политиками конфиденциальности ВКонтакте. Нужно предусмотреть работу в рамках открытых API и ограничений.
* Точность алгоритмов: Сложность может заключаться в точности определения схожести интересов, так как интересы и увлечения могут быть выражены в различных формах (например, текст, картинки, группы).
* Ограничения на использование персональных данных: Сервис должен соблюдать законы о защите персональных данных, что может потребовать анонимизации и получения согласий на использование определенной информации.
* Обработка больших объемов данных: Необходимо учитывать масштабируемость системы, так как пользователи могут быть тысячами или миллионами.
* Производительность и скорость: Алгоритмы, работающие с большими массивами данных, должны быть достаточно быстрыми для оперативного поиска и рекомендаций.

### Технические требования:

* Интеграция с ВКонтакте: Система должна эффективно взаимодействовать с API социальной сети для сбора данных о пользователях и их интересах.
* База данных: Необходимо хранение больших объемов данных о пользователях, интересах и результатах поиска.
* Обработка и анализ данных: Алгоритмы должны поддерживать высокую производительность при обработке и анализе данных пользователей.

# Глава 3.

## 3.1 Теоретическое обоснование проекта.

Алгоритм определения возраста в этой части построен на использовании социальных графов и векторных моделей для распространения меток:

### Этапы алгоритма:

* Инициализация:
  + Узлы-пользователи получают метки на основе разметки (известные значения возраста).
* Построение векторной модели:
  + Modelavg: Среднее распределение значений атрибута (возраста) для соседей узлов-пользователей. Используется для распространения меток между пользователями.
  + Modelmax: Вероятность для соседей иметь идентичные значения атрибута. Используется для распространения меток в сторону сообществ.
* Распространение меток:
  + Метки распространяются:
    - Сначала от пользователей к сообществам.
    - Затем обратно от пользователей и сообществ к пользователям, у которых метки отсутствуют.
* Оценка близости распределений:
  + Modelavg: Используется для более гибкого распространения меток.
  + Modelmax: Максимальная вероятность достигается, когда все соседи имеют одинаковое значение атрибута.



Изображение выглядит как Шрифт, текст, белый, линия

Автоматически созданное описание

Здесь S — близость, соответствующая максимально близкому значению атрибута val, Distr — распределение значений атрибута соседей, Model\* (val) — распределение из модели для значения атрибута val

Для определения степени схожести пользователей используются следующий набор параметров: город, возраст (указанный в профиле), предсказанный возраст, общее количество друзей, количество общих друзей, список интересов (тематики групп пользователя).

Далее рассчитываются значения схожести *[0;1]* для каждого из параметров:

1. Город: значение схожести для этого параметра строго равен 1, если значения параметра полностью совпадают для двух пользователей, в противном случае, равен 0;

2. Возраст (указанный в профиле) и предсказанный: значения схожести рассчитываются по формуле:

Где – возраст первого пользователя.

– возраст второго пользователя.

– абсолютная разница в возрасте между пользователями.

Общие друзья: значение схожести рассчитывается по формулам:

Где – количество друзей первого пользователя.

– количество друзей второго пользователя.

– четверть от наименьшего количества друзей между двумя пользователями.

– количество общих друзей

Значение схожести по интересам рассчитывается по формуле:

Где – множество интересов (тематик) первого пользователя.

– множество интересов (тематик) второго пользователя.

После определения значений схожестей каждого из параметров, рассчитывается общая схожесть пользователей. Для этого определяются динамические веса, в зависимости от наличия того или иного набора параметров при расчётах, описанных ранее.

Значения весов для параметров находятся в следующих диапазонах: Город: [0.1, 0.3]; Возраст: [0.1, 0.3]; Предсказанный возраст: [0, 0.25]; Общие друзья: [0.1, 0.3]; Город: [0.1, 0.3].

Итоговая схожесть пользователь рассчитывается следующим образом:

Где – количество переданных параметров.

– вес -го параметра.

– значение схожести -го параметра.

Результат нормализован по сумме всех весов.

## 3.2 Практическое обоснование проекта.

В основе проекта лежит использование стороннего API (VK API) для получения данных о пользователях. Для доступа к некоторым методам VK API, требовалась авторизация, которую необходимо было реализовать как на фронт, так и на бэк-части приложения. Отправка запросов на авторизацию была частично реализована с помощью JavaScript библиотеки VK SDK, другие запросы реализованы через библиотеку JQuery.

Бэкенд-часть приложения построена на C# фреймворке .NET Core, помимо этого, использовался Entity Framework 8.0, для взаимодействия с БД PostgreSQL из кода C#.

Также, к бэкенд-части приложения относятся модули Python, отвечающие за параллельные расчёты предполагаемого возраста и сходства пользователей. После получения всех необходимых данных, из кода C# запускаются несколько потоков интерпретаторов Python, тем самым, позволяя быстрее (по сравнению с последовательными расчётами) обработать данные.

## 3.3 Собственные практические решения.

В данном разделе рассматриваются ключевые подходы и решения, которые были разработаны и реализованы в рамках проекта по созданию сервиса для нахождения друзей в социальной сети ВКонтакте. Основное внимание уделено интеграции алгоритмов поиска, использующих схожесть социальных цифровых портретов, которые учитывают как демографические параметры пользователей, так и их интересы и увлечения.

Описание методов

Выполнение входа по VKID – для каждого пользователя, бэкенд генерирует уникальный идентификатор сессии и набор PKCE (Proof Key for Code Exchange), после чего, отправляет их клиенту на фронт. Далее, пользователь может авторизоваться через VKID, используя соответствующую кнопку.

*Функция создания интерфейса (кнопки) авторизации VKID*

function InitVKIDOneClick(state, codeChallenge) {

if ('VKIDSDK' in window) {

const VKID = window.VKIDSDK;

//Инициализация VK Api SDK

VKID.Config.init({

app: 51919588,

redirectUrl: 'https://byadminpresents.github.io/TestVKService/',

responseMode: VKID.ConfigResponseMode.Callback,

source: VKID.ConfigSource.LOWCODE,

scope: '',

codeChallenge,

state

});

//Создание OneTap кнопки авторизации VKID

const oneTap = new VKID.OneTap();

oneTap.render({

container: mainElement,

showAlternativeLogin: true

})

.on(VKID.WidgetEvents.ERROR, vkidOnError)

//Получение кода доступа и device\_id для дальнейшей авторизации

.on(VKID.OneTapInternalEvents.LOGIN\_SUCCESS, function (payload) {

const code = payload.code;

const deviceId = payload.device\_id;

fetch("https://{backend\_endpoint\_address}/VKID/PerformVKIDLogin?code=" + code + "&deviceId=" + deviceId, {

method: 'POST',

credentials: 'include' // Отправлять cookie с запросом

})

.then((response) => {

if (!response.ok) {

throw new Error(`HTTP err: ${response.status}`);

}

return response.json();

})

.then((data) => {

console.log(data)

})

.catch((error) => {

console.error("err:", error);

});

sendPostRequest(code, deviceId, state)

});

}

}

После получения кода авторизации клиент отправляет их на бэкенд, для дальнейшего запроса и сохранения токенов авторизации VKID.

*Метод получения токенов VKID с бэкенда приложения*

public static async Task<VKIDTokensHelperModel?> VKIDLogin(string code, string deviceId, string codeVerifier, string state)

{

string url = "https://id.vk.com/oauth2/auth";

//Объявление параметров запроса

var parameters = new[]

{

new KeyValuePair<string, string>("grant\_type", "authorization\_code"),

new KeyValuePair<string, string>("client\_id", "51919588"),

new KeyValuePair<string, string>("device\_id", deviceId),

new KeyValuePair<string, string>("redirect\_uri", "https://byadminpresents.github.io/TestVKService/"),

new KeyValuePair<string, string>("code\_verifier", codeVerifier),

new KeyValuePair<string, string>("code", code),

new KeyValuePair<string, string>("state", state)

};

string json = System.Text.Json.JsonSerializer.Serialize(parameters);

//Инициализация HttpClient и отправка запроса

using (HttpClient client = new HttpClient())

{

var content = new FormUrlEncodedContent(parameters);

HttpResponseMessage response = await client.PostAsync(url, content);

var token =await response.Content.ReadAsAsync<VKIDTokensHelperModel>();

return token;

}

}

Указание параметров и запуск поиска – происходят на фронт части приложения.

Пользователю доступны для настройки определенные параметры, такие как: ФИО, пол,

возраст, город проживания, университет, семейное положение и др.

При нажатии на кнопку поиска клиент отправляет соответствующий запрос на бэкенд.

Далее, происходит проверка статуса авторизации пользователя, а затем, запрос и кэширование данных пользователей:

[HttpPost("CompareUsers")]

public ActionResult CompareUsers()

{

var cookies = HttpContext.Request.Cookies;

VKIDToken? VKIDToken = null;

var queryDict = HttpContext.Request.Query.Select(x => new { key = x.Key.ToString(), value = x.Value.ToString() }).ToDictionary(x => x.key, x => x.value);

if (cookies.TryGetValue("state", out var state))

{

return DataHandler.SearchAndCompareUsers(db, state, queryDict);

}

return new JsonResult("You're not logged in");

}

Для расчёта предполагаемого возраста был разработан модуль python:

Метод *calculate\_distribution* рассчитывает распределение возрастов среди соседей пользователя (друзей и подписок). Результат представляет собой словарь, где ключи - это возраста, а значения - количество людей данного возраста.

def calculate\_distribution(self, userAges: List[int]) -> Dict[int, int]:

age\_distribution = defaultdict(int)

for age in userAges:

if age is not None:

age\_distribution[age] += 1 # Увеличиваем счетчик для возраста

return age\_distribution

Описание:

* Принимает на вход массив возрастов пользователей userAges.
* Проверяет наличие значения возраста и добавляет его в распределение.
* Возвращает итоговый словарь возрастов.

Метод *cosine\_similarity* вычисляет косинусное сходство между двумя векторами, представляющими распределения возрастов.

def cosine\_similarity(self, vec1: List[int], vec2: List[int]) -> float:

vec1 = np.array(vec1).reshape(1, -1)

vec2 = np.array(vec2).reshape(1, -1)

return cosine\_similarity(vec1, vec2)[0][0] # Возвращаем значение сходства

Описание:

* Использует библиотеку sklearn для расчета косинусного сходства.
* Преобразует входные списки в двумерные массивы numpy для совместимости с cosine\_similarity.
* Возвращает значение косинусного сходства, которое показывает степень схожести двух векторов.

Метод predict\_age\_for\_user прогнозирует возраст пользователя на основе возрастного распределения его друзей и подписок. Если возраст пользователя уже указан, он возвращает его.

def predict\_age\_for\_user(self, user\_id: int) -> Optional[int]:

user\_data = self.get\_user\_data\_from\_vk(user\_id)

if user\_data.get('age') is not None:

return user\_data['age'] # Если возраст уже указан, возвращаем его.

# Получаем соседей (друзья пользователя)

friend\_ids = self.get\_user\_friends(user\_id)

subscription\_ids = self.get\_user\_subscriptions(user\_id)

# Рассчитываем распределения возрастов для друзей и подписчиков групп

friend\_age\_distribution = self.calculate\_distribution(friend\_ids)

subscription\_age\_distribution = self.calculate\_distribution(subscription\_ids)

# Создаем векторы распределений для возрастов от 17 до 100 лет

max\_age = 100

friend\_age\_vector = [friend\_age\_distribution.get(age, 0) for age in range(17, max\_age + 1)]

subscription\_age\_vector = [subscription\_age\_distribution.get(age, 0) for age in range(17, max\_age + 1)]

similarity\_friend = self.cosine\_similarity(friend\_age\_vector, friend\_age\_vector)

similarity\_subscription=self.cosine\_similarity(subscription\_age\_vector, subscription\_age\_vector)

weighted\_friend\_distribution = [x \* similarity\_friend \* self.w\_user for x in friend\_age\_vector]

weighted\_subscription\_distribution = [x \* similarity\_subscription \* self.w\_comm for x in subscription\_age\_vector]

total\_distribution=np.add(weighted\_friend\_distribution, weighted\_subscription\_distribution)

# Нормализуем результат

total\_distribution = total\_distribution / np.sum(total\_distribution)

# Предсказываем возраст на основе распределений (max)

predicted\_age = np.argmax(total\_distribution) + 17

return predicted\_age

1. Данный метод собирает ID друзей и подписок, вычисляет возрастное распределение для каждой категории с помощью calculate\_distribution.

2. Формирует векторы распределений возрастов для диапазона 17-100 лет.

3. Вычисляет косинусное сходство между распределениями для друзей и подписок.

4. Взвешивает распределения, используя заданные веса (w\_user, w\_comm) и значения сходства.

5. Нормализует итоговое распределение и выбирает возраст с максимальной вероятностью.

## 3.4 Особенности разработки.

Разработка подобного решения, предполагающего обращение к стороннему высоконагруженному API с целью получения большого количества данных, требует тщательного изучения возможностей и ограничений стороннего сервиса.

В рамках одного запроса поиска происходит сбор данных около полумиллиона пользователей. Получить такой объём информации за один раз от стороннего API не представляется возможным ввиду ограничений на количество запрашиваемых пользователей в рамках одного запроса (1000 профилей).

С другой стороны, сервис позволяет объединить несколько запросов в один (до 25 подзапросов), тем самым увеличивая количество возможной получаемой информации до 25000 пользователей. Помимо этого, запросы могут быть отправлены параллельно, но не более 5 в секунду (при доступе в VK API по сервисному ключу). Подобный подход позволяет получать информацию о 125,000 пользователях в секунду (без учёта времени на получение ответа и прочих задержек сети).

## 3.5 Функциональные возможности проекта.

Функциональными возможностями проекта является параметризация поиска, определение и схожести пользователей на основе интересов и предсказанного возраста, полученных в ходе анализа профилей и прочей информации, определение предполагаемого возраста человека.

## 3.6 Пользовательский интерфейс.

Пользовательский интерфейс представляет из себя веб-страницу с блоками полей фильтра поиска, найденных пользователей и информации о выбранном пользователе.

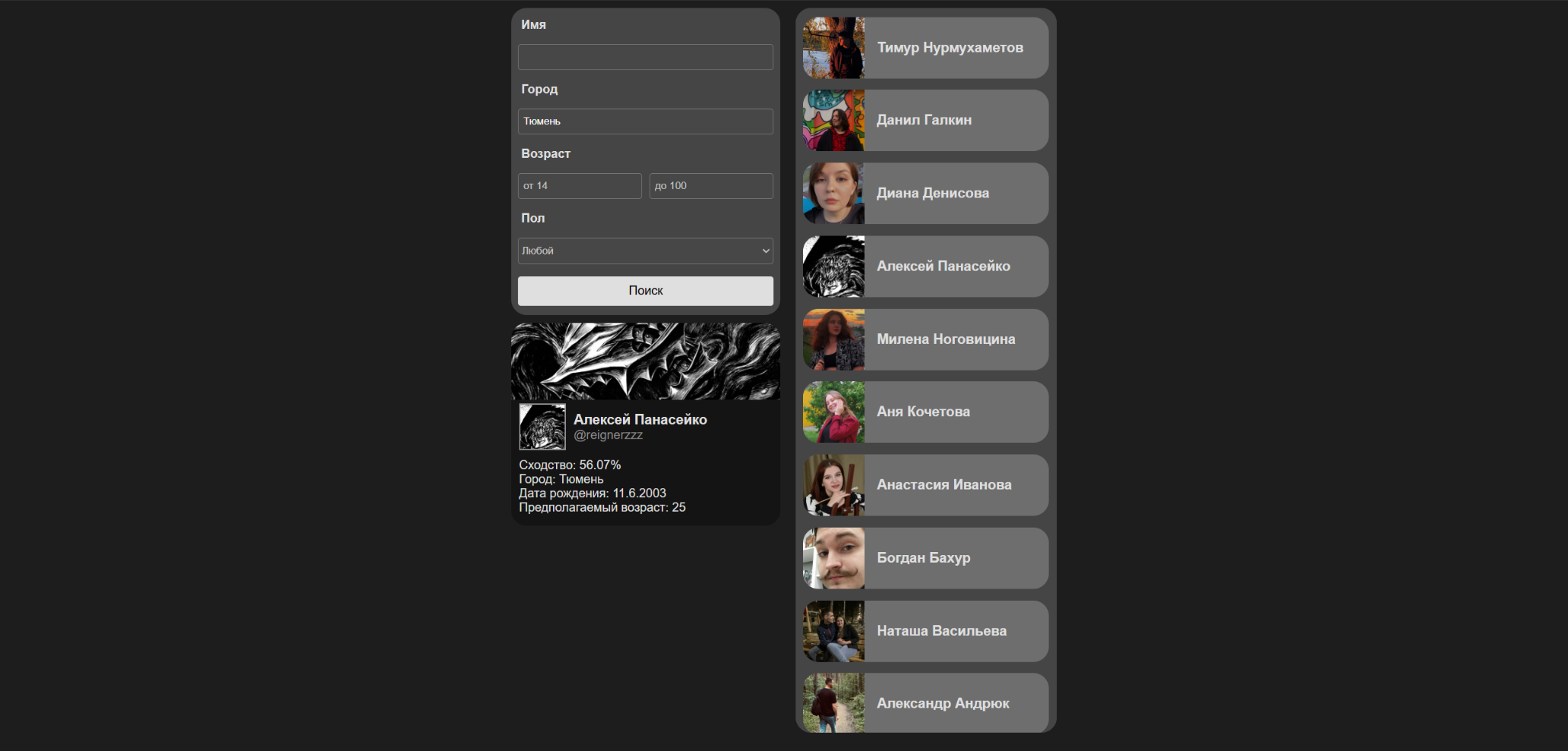


Рис.1 Страница сервиса

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис.2 Блок фильтров поиска

Изображение выглядит как Человеческое лицо, снимок экрана, текст, человек

Автоматически созданное описание

Рис.3 Блок результатов поиска

Изображение выглядит как текст, снимок экрана, снег, дерево

Автоматически созданное описание

Рис.4 Блок информации о выбранном пользователе

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рис.5 Всплывающее уведомление авторизации

# Список используемой литературы

1. Гомзин, А.Г., Кузнецов С.Д. Метод автоматического определения

возраста пользователей с помощью социальных связей [Электронный ресурс]: исследовательская работа – Электрон. текстовые дан. Труды ИСП РАН, том 28, 2016 г. стр. 171-184. DOI: 10.15514/ISPRAS-2016-28(6)-12. Веб-ресурс:https://www.researchgate.net/publication/312201357\_A\_method\_of\_automatically\_estimating\_user\_age\_using\_social\_connections/fulltext/587630f808ae8fce492dacb6/A-method-of-automatically-estimating-user-age-using-social-connections.pdf