TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа ядерных технологий
Направление подготовки – 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) – Отделение экспериментальной физики

БАКАЛАВРСКАЯ РАБОТА

Тема работы
Разработка рекомендательного бота для социального взаимодействия

УДК 004.75:004.773.6:004.8

Студенты

Группа	ФИО	Подпись	Дата
0В8Б	Брылин Артем Владимирович		

Руководитель ВКР

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭФ	Семенов Михаил Евгеньевич	Кандидат ф. – м. наук, доцент		

Со-руководитель (по разделу «Концепция стартап-проекта»)

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ШИП	Селевич Ольга Семеновна	Кандидат экономических наук, доцент		

КОНСУЛЬТАНТЫ:

По разделу «Социальная ответственность»

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ООД ШБИП	Сечин Андрей Александрович	Кандидат технических наук, доцент		

ДОПУСТИТЬ К ЗАЩИТЕ:

Руководитель ООП	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭФ ИЯТШ	Крицкий Олег Леонидович	Кандидат ф.– м. наук, доцент		

Томск – 2022 г.

TOMSK POLYTECHNIC UNIVERSITY ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский Томский политехнический университет» (ТПУ)

Школа – Инженерная школа ядерных технологий
Направление подготовки – 01.03.02 Прикладная математика и информатика
Отделение школы (НОЦ) – Отделение экспериментальной физики

УТВЕРЖДАЮ:
Руководитель ООП
_____ _____ Крицкий О.Л.
(Подпись) (Дата) (Ф.И.О.)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

В форме:

Бакалаврской работы

Студенту:

Группа	ФИО
0В8Б	Брылину Артему Владимировичу

Тема работы:

Разработка рекомендательного бота для социального взаимодействия	
Утверждена приказом директора (дата, номер)	

Срок сдачи студентом выполненной работы:	

ТЕХНИЧЕСКОЕ ЗАДАНИЕ:

Исходные данные к работе	
Перечень подлежащих исследованию, проектированию и разработке вопросов	1. Провести сравнительный анализ схожих по тематике ботов и приложений по теме работы. 2. Разработать алгоритм работы и архитектуру бота для осуществления взаимодействия между пользователями и ботом. 3. Разработать техническую документацию. 4. Реализация бота на выбранном языке программирования. 5. Подготовка и проведение тестирования программы.
Перечень графического материала	1. Диаграммы алгоритмов работы программного продукта. 2. UML диаграммы архитектуры программного продукта.

	3. Скриншоты работы программного продукта. 4. Визуализация взаимодействия с ботом. 5. Визуализация работоспособности бота с пользователями.
Консультанты по разделам выпускной квалификационной работы	
Раздел	**Консультант**
Концепция стартап-проекта	Селевич Ольга Семеновна, доцент ШИП
Социальная ответственность	Сечин Андрей Александрович, доцент ООД ШБИП

Дата выдачи задания на выполнение выпускной квалификационной работы по линейному графику	

Задание выдал руководитель:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент ОЭФ ИЯТШ	Семенов Михаил Евгеньевич	к. ф.-м. н., доцент		

Задание приняли к исполнению студент:

Группа	ФИО	Подпись	Дата
0В8Б	Брылин Артем Владимирович		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«КОНЦЕПЦИЯ СТАРТАП-ПРОЕКТА»

Студенту:

Группа	ФИО
0В8Б	Брылин Артем Владимирович

Школа	**ИЯТШ**	**Направление**	01.03.02 Прикладная математика и информатика
Уровень образования	бакалавриат		

Перечень вопросов, подлежащих разработке:	
Проблема конечного потребителя, которую решает продукт, который создается в результате выполнения НИОКР (функциональное назначение, основные потребительские качества)	Продукт позволяет его потребителям взаимодействовать с другими людьми, тем самым социализируя пользователей.
Способы защиты интеллектуальной собственности	Регистрация товарных знаков и защита авторских прав на программу ЭВМ и базу данных
Объем и емкость рынка	Объем российского рынка составляет от 70 млн $.
Современное состояние и перспективы отрасли, к которой принадлежит представленный в ВКР продукт	Рынок активно развивается. Количество потребителей постоянно возрастает.
Себестоимость продукта	Оценена в 299000 рублей
Конкурентные преимущества создаваемого продукта	Удобный интерфейс, понятный дизайн, популярная используемая платформа продвижения, более персонализированные настройки пользователя.
Сравнение технико-экономических характеристик продукта с отечественными и мировыми аналогами	Наличие тэговой системы и геймификация элементов графического пользовательского интерфейса
Целевые сегменты потребителей создаваемого продукта	Мужчины и женщины в возрасте от 16 до 45 лет.

Бизнес-модель проекта	Монетизация проекта будет производиться путем рекламы.
Каналы продвижения продукта	Интернет, социальные сети, мессенджеры
Перечень графического материала:	
При необходимости представить эскизные графические материалы (например, бизнес-модель)	Бизнес – модель Остервальдера - Пинье

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант по разделу «Концепция стартап-проекта» (со-руководитель ВКР):

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Доцент	Селевич Ольга Семеновна	к.э.н.		

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
0В8Б	**Брылин Артем Владимирович**		

ЗАДАНИЕ ДЛЯ РАЗДЕЛА
«СОЦИАЛЬНАЯ ОТВЕТСТВЕННОСТЬ»

Студенту:

Группа		ФИО	
0В8Б		Брылин Артем Владимирович	
Школа	Инженерная школа ядерных технологий	**Отделение (НОЦ)**	Отделение экспериментальной физики
Уровень образования	Бакалавриат	**Направление/ специальность**	01.03.02 Прикладная математика и информатика

Тема ВКР:

Разработка рекомендательного бота для социального взаимодействия	
Исходные данные к разделу «Социальная ответственность»:	
Введение – Характеристика объекта исследования (вещество, материал, прибор, алгоритм, методика) и области его применения. – Описание рабочей зоны (рабочего места) при разработке проектного решения/при эксплуатации	*Объект исследования:* бот для мессенджера Telegram *Область применения:* Интернет *Рабочая зона:* жилое помещение *Размеры помещения:* 4*6 м. *Количество и наименование оборудования рабочей зоны:* Стол, стул, компьютер, роутер. *Рабочие процессы, связанные с объектом исследования, осуществляющиеся в рабочей зоне:* программирование бота на компьютере, тестирование бота на компьютере
Перечень вопросов, подлежащих исследованию, проектированию и разработке:	
1. Правовые и организационные вопросы обеспечения безопасности при разработке проектного решения/при эксплуатации – специальные (характерные при эксплуатации объекта исследования, проектируемой рабочей зоны) правовые нормы трудового законодательства; – организационные мероприятия при компоновке рабочей зоны.	Федеральный закон от 28 декабря 2013 г. N 426-ФЗ "О специальной оценке условий труда; ТК РФ от 30.12.2001 N 197-ФЗ (ред. от 01.04.2019); ГОСТ 12.2.032-78 «ССБТ. Рабочее место при выполнении работ сидя. Общие эргономические требования».
2. Производственная безопасность при разработке проектного решения/при эксплуатации – Анализ выявленных вредных и опасных производственных факторов – Обоснование мероприятий по снижению воздействия	**Опасные факторы:** 1. Производственные факторы, связанные с электрическим током 2. Повышенная температура рабочей поверхности 3. Электрический ток **Вредные факторы:** 1. Повышенный уровень электромагнитных излучений 2. Недостаточная освещенность рабочей зоны 3. Нервно-психические перезагрузки 4. Отклонения микроклимата
3. Экологическая безопасность при разработке проектного решения/при эксплуатации	**Воздействие на селитебную зону:** Повышенная нагрузка на сеть интернет **Воздействие на литосферу:** Отсутствует

	Воздействие на гидросферу: Отсутствует **Воздействие на атмосферу:** Повышение температуры рабочей зоны
4. Безопасность в чрезвычайных ситуациях при разработке проектного решения/при эксплуатации	**Возможные ЧС:** Пожар, короткое замыкание **Наиболее типичная ЧС:** Существует вероятность возникновения пожаров
Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант:

Должность	ФИО	Ученая степень, звание	Подпись	Дата
Доцент	Сечин Андрей Александрович	к.т.н.		

Задание принял к исполнению студент:

Группа	ФИО	Подпись	Дата
0В8Б	Брылин Артем Владимирович		

Реферат

Выпускная квалификационная работа содержит 98 страниц, 45 рисунков, 9 таблиц, 30 источников, 1 приложение.

Ключевые слова: чат-бот, мобильные приложения, приложения для знакомств, дескрипторы, информационные системы.

Цель работы – проектирование и реализация социального чат-бота на основе тэговой системы и алгоритмов поиска профилей.

В результате работы был разработан и программно реализован алгоритм для работы с социальным ботов, была предложена рекомендательная система на основании тэгов, используемая в боте для формирования очереди профилей.

Для разработки бота был использован язык программирования Python, для хранения данных была выбрана СУБД PostgreSQL с использованием библиотеки Psycopg2, также была использована бибилотека pyTelegramBotAPI.

Определения, обозначения, сокращения, нормативные ссылки

В данной работе применены следующие термины с соответствующими определениями:

Чат-бот – «это робот, предназначенный для осуществления коммуникаций с пользователями в сети Интернет, выполняющий действия согласно заложенному сценарию» (22).

В данной работе использованы следующие сокращения:

ИКТ – информационно-коммуникативные технологии.

ИИ – искусственный интеллект.

API – прикладной программный интерфейс.

Python – высокоуровневый язык программирования.

БД – база данных.

СУБД – система управления базами данных.

PostgreSQL – свободная объектно-реляционная система управления базами данных.

1. Оглавление

Введение ... 8

1. Выбор платформы для реализации приложения ... 10

1.1. Логическая модель и основные этапы работы бота ... 12

1.2. Тэговая система рекомендаций ... 17

2. Разработка базы данных ... 20

2.1. Концептуальное проектирование ... 20

2.2. Построение даталогической модели ... 22

2.3. Реализация базы данных ... 23

2.4. Выбор СУБД ... 23

3. Разработка алгоритма и программы ... 25

3.1. Выбранные технологии разработки ... 25

3.2. Реализация и описание алгоритмов ... 25

3.4. Регистрация пользователя ... 28

3.5. Главное меню ... 28

3.6. Поиск профилей других пользователей ... 29

3.7. Изменение своего профиля ... 29

3.8. Выключение профиля ... 29

4. Задание для раздела «Концепция стартап-проекта» ... 30

4.1. Проблема конечного потребителя ... 32

4.2. Способы защиты интеллектуальной собственности ... 33

4.3. Объем и емкость рынка ... 33

4.4. Современное состояние и перспективы отрасли, к которой принадлежит представленный в ВКР продукт ... 34

4.5. Себестоимость продукта ... 34

4.6. Конкурентные преимущества создаваемого продукта ... 35

4.7. Сравнение технико-экономических характеристик продукта с отечественными и мировыми аналогами ... 38

4.8. Целевые сегменты потребителей создаваемого продукта ... 40

4.9. Бизнес-модель проекта ... 42

4.10. Каналы продвижения продукта ... 43

5. Социальная ответственность ... 45

5.1. Введение ... 45

5.2. Правовые и организационные вопросы обеспечения безопасности ... 45

5.3. Производственная безопасность ... 46

5.4. Экологическая безопасность ... 51

5.5. Безопасность в чрезвычайных ситуациях 53
5.6. Выводы по разделу «Социальная ответственность» 54
Заключение 55
Список публикаций студента 56
Список использованных источников 57
Приложение А 61

Введение

В современном обществе глобальную роль стала играть информация, способы и средства ее распространения. Личные и социальные коммуникации с целью развлечения и делового сотрудничества все чаще переносятся в сети Интернет и распространяются при помощи ИКТ. Исследователи определяют современное общество как «информационное общество», «общество знания», «сетевое общество», «общество риска» [7],[8],[21].

В XXI веке чрезвычайную актуальность приобрела разработка социальных чат-ботов как «автоматизированных программных комплексов, позволяющих распространять информацию с большой скоростью и эффективностью, в том числе за счет имитации поведения реальных пользователей социальных сетей» [7, с. 20].

Под чат-ботом (to chat – непринужденный разговор в сети Интернет, bot – робот) будем понимать компьютерную программу (робота), предназначенную для осуществления коммуникаций с пользователями в сети Интернет, по определенному сценарию поведения [22]. Содержательно чат-бот это программа, которая может общаться с пользователями (имитируя поведение человека) для достижения какой-либо цели или развлечения. Такая коммуникация может быть выстроена через платформу обмена сообщениями, например: Facebook Messenger, Slack, Telegram, Viber», «ВКонтакте» или «Одноклассники» [3].

В настоящее время по способу взаимодействия с пользователем выделяют два типа чат-ботов: *кнопочные* и *текстовые*. Кнопочные чат-боты основаны на использовании относительно простых программ, основанных на правилах, в то время как текстовые могут использовать ИИ и пригодны для решения более широкого круга задач. По функциональному назначению отметим *финансовых* ботов (предоставляют курс валют, стоимость различных акций), *рекомендательных* (мониторинг цен на товары, список рекомендаций фильмов для пользователей по их предпочтениям), а также *социальных*, которые позволяют людям взаимодействовать между собой.

В рамках данной работы мы спроектировали и программно реализовали социального бота, который с помощью указываемых пользователем дескрипторов (тэгов) позволял находить людей со схожими предпочтениями и обменивать их контактной информацией.

Цель работы - проектирование и реализация социального чат-бота на основе тэговой системы и алгоритмов поиска профилей. Для выполнения цели работы ставятся следующие **задачи**:

- Разработать логику работы социального чат-бота
- Выбрать стек технологий для разработки
- Реализовать тэговую систему и алгоритмы
- Провести тестирование работоспособности

1. Выбор платформы для реализации приложения

Первые чат-боты появились задолго до появления мессенджеров и социальных сетей, мобильных телефонов и персональных компьютеров.

В 1941 году английский математик Алан Тьюринг начал заниматься теорией машинного интеллекта (прообразом искусственного интеллекта) с целью определить может ли машина мыслить, как человек. Суть теста Тьюринга состояла в задавании вопросов машине и получении печатных ответов.

В 1966 году в Массачусетском технологическом институте была создана революционная программа-бот ELIZA, которая стала виртуальным собеседником на основе обработки естественного языка. ELIZA имитировала диалог с психотерапевтом и хотя она еще не обладала искусственным интеллектом, на основе использования слов-якорей она строила вопросы и была способна вести диалог с помощью телепринтера еще за 15 лет до появления компьютера. Позже появились другие виртуальные собеседники: Студент (1964), Шрдлу (1970), PARRY (1972). Однако, настоящая революция произошла в 1995 году, когда на основе ELIZA была создана программа A.L.I.C.E. На протяжение более 20 лет она оставалась самым совершенным виртуальным собеседником, способным поддерживать диалог более, чем на 40000 различных тем. До сих пор на базе A.L.I.C.E. создаются современные чат-боты.

В середине 2010-х годов использование ботов на основе их интеграции с мессенджерами и соцсетями получило широкое распространение, боты стали одним из трендов в интернет-маркетинге [1]. Авторы [26, 27] выделяют четыре основные сферы использования ботов: бизнес/коммерция; получение информации (СМИ); образование; развлечение.

В настоящее время социальная коммуникация с использованием чат-ботов реализована в социальных сетеях и мессенджерах Facebook Messenger, Slack, Telegram, Viber», «ВКонтакте», «Одноклассники» [3], а также порталах, например, gosuslugiu.ru. Профессиональная стоимость разработки чат-бота начинается от 20 000 рублей.

Таблица 1 – Сравнение популярнейших мессенджеров

Пункт сравнения	Telegram	Viber	WhatsApp
Сообщения и медиа	Текстовые и голосовые сообщения, документы до 2 ГБ	Текстовые и голосовые сообщения, документы до 200 МБ	Текстовые и голосовые сообщения, документы до 100 МБ
Групповые чаты	Беседы до 200000 пользователей	Беседы до 250 пользователей	Беседы до 256 человек
Безопасность	Сквозное шифрование. Исчезающие сообщения. Возможность удаления чата у обоих участников одним	Сквозное шифрование. Исчезающие сообщения. Удаление сообщений	Сквозное шифрование. Удаление сообщений с оставлением пометки удалено
Аудитория, млн	37.88	34.84	76.83
Наличие ботов	Присутствует	Присутствует	Присутствует

В 2021 году Telegram обогнал по количеству ежемесячных пользователей Viber, тем самым став вторым самым популярным мессенджером в России. Его аудитория насчитывает более 37 млн. активных пользователей каждый месяц (рисунок 1). Telegram – мессенджер, позволяющий пользователям обмениваться текстовыми и медиа-сообщениями между собой. Telegram является кроссплатформенным – он существует на Windows, iOS, Android и других операционных системах. Функционал Telegram очень разнообразный и постоянно расширяется. В мессенджере реализована возможность обмена обычными файлами и медиафайлами, текстовыми и голосовыми сообщениями. Все сообщения хранятся на серверах Telegram, тем самым позволяя пользователям хранить всю информацию не только у себя на устройстве. В Telegram можно вести каналы, а также создавать и использовать ботов с помощью прикладного программного интерфейса API Telegram. Функционал ботов может быть очень разнообразным и ограничен только API Telegram и задумкой разработчиков.

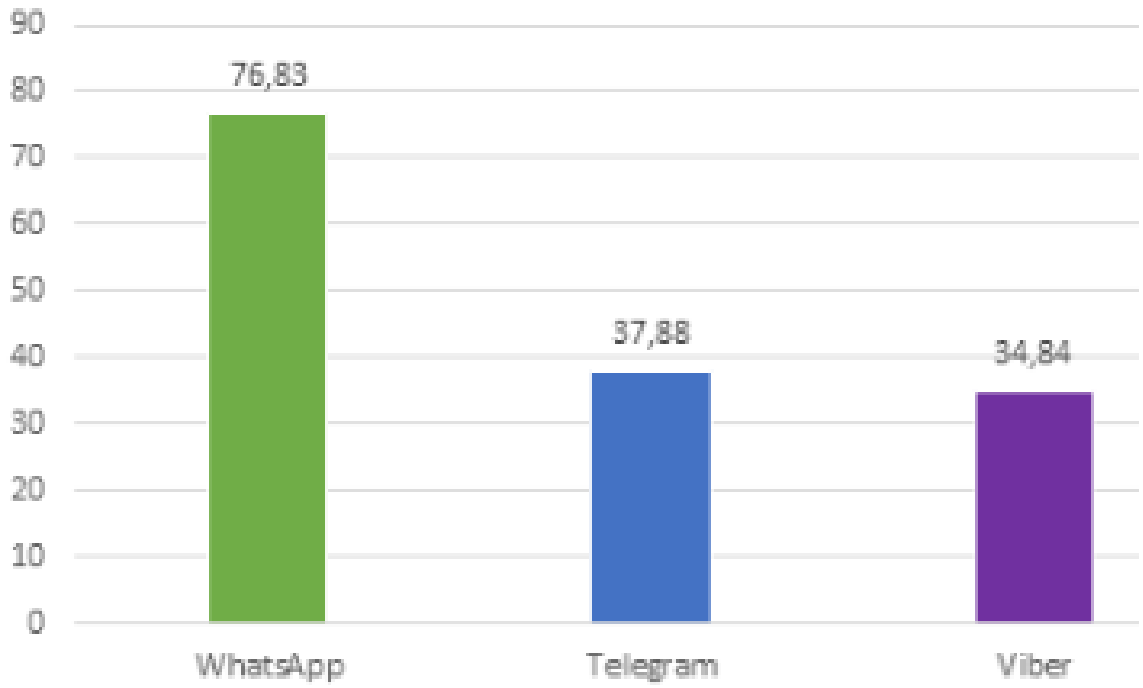


Рисунок 1 – Количество пользователей в крупнейших мессенджерах, млн. человек в месяц, Россия, 2021 год

Опираясь на характеристики мессенджеров (таблица 1), сделаем следующие выводы:

- функционал Telegram значительно больше и разнообразней чем у его ближайших конкурентов (например, видео-сообщения и аудио-сообщений с просмотром в ускоренном режиме);
- аудитория Telegram более молодая, чем у WhatsApp и Viber. Молодежь от 18 до 24 лет чаще использует Telegram чем его конкурентов.
- Молодежь чаще использует функционал ботов.
- Telegram имеет хорошо документированное описание API для разработчиков.

Для нашей разработки мы выбрали Telegram в качестве платформы для реализации социального чат-бота.

1.1. Логическая модель и основные этапы работы бота

На рисунке 2 представлена логика взаимодействия пользователя с ботом, которая включает два основных этапа: регистрацию и непосредственно

социальное взаимодействие – поиск профилей, редактирование и выключение профиля.

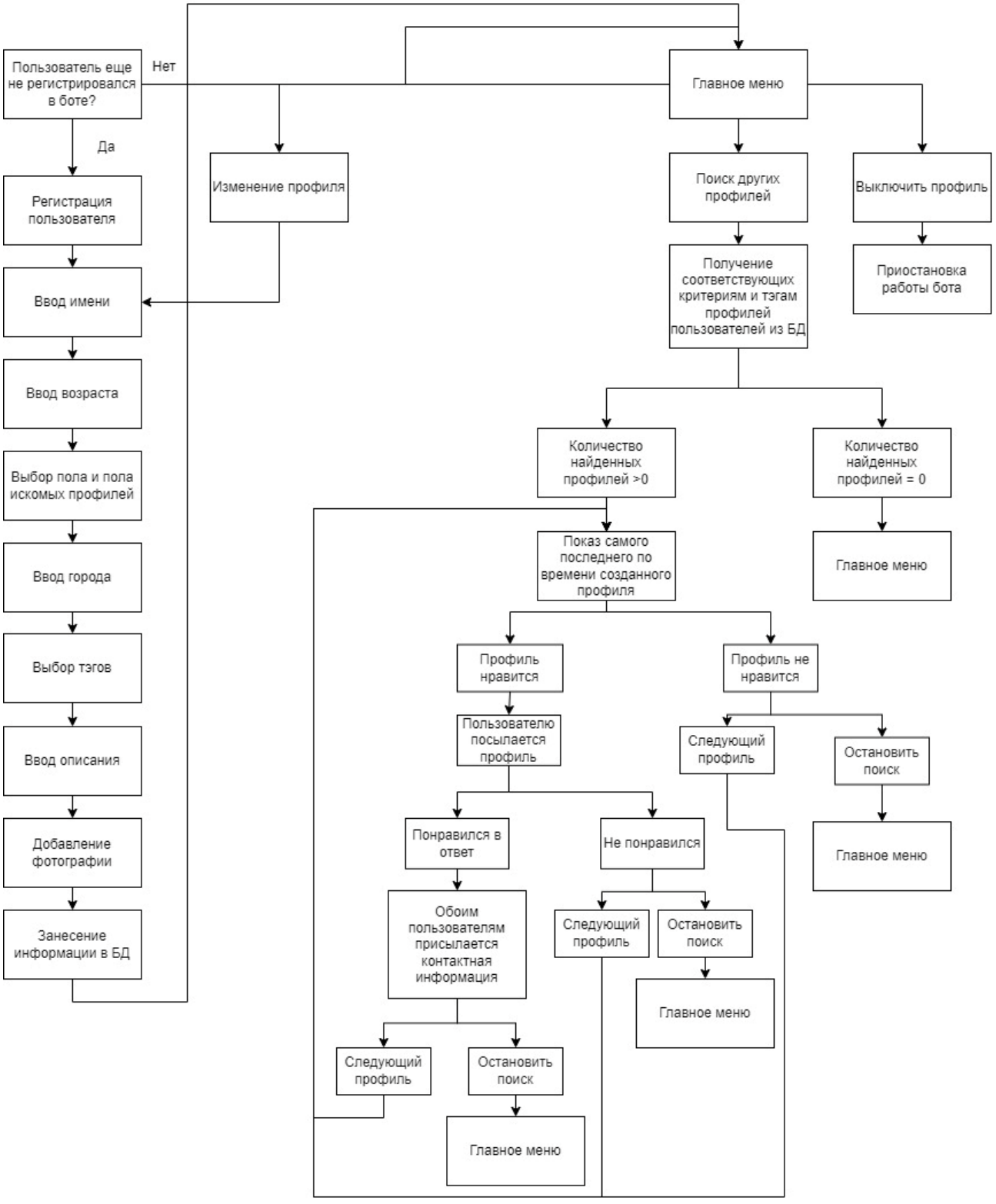


Рисунок 2 – Этап регистрации пользователя в боте (слева) и основная работа (справа)

Регистрация представляет из себя поочередный ввод данных, необходимых для формирования профиля (имя, возраст, пол, город и т.д.). Примеры графического интерфейса поиск профилей других пользователей, редактирования и выключения профиля приведены на рисунках 13-19.

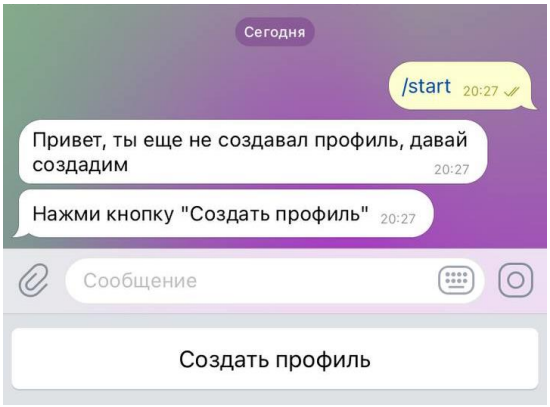


Рисунок 3 – Запуска бота и регистрации пользователя

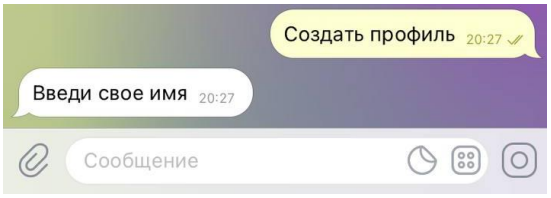


Рисунок 4 – Ввод имени пользователя

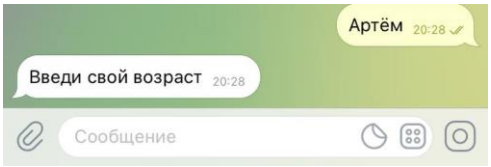


Рисунок 5 - Ввод возраста пользователя

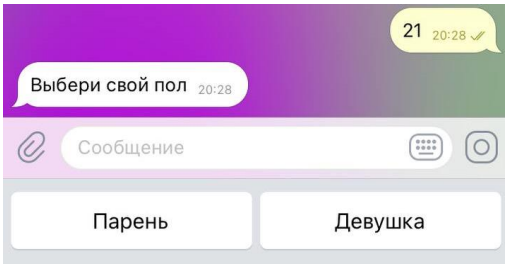


Рисунок 6 – Выбор пола пользователя

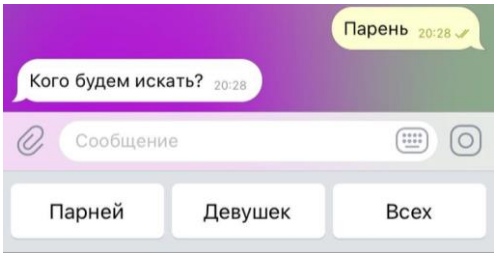


Рисунок 7 – Выбор пола для поиска пользователей

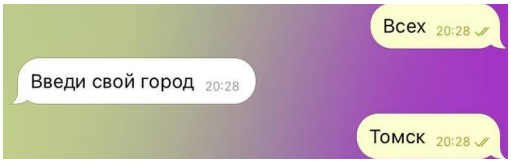


Рисунок 8 – Ввод города пользователем

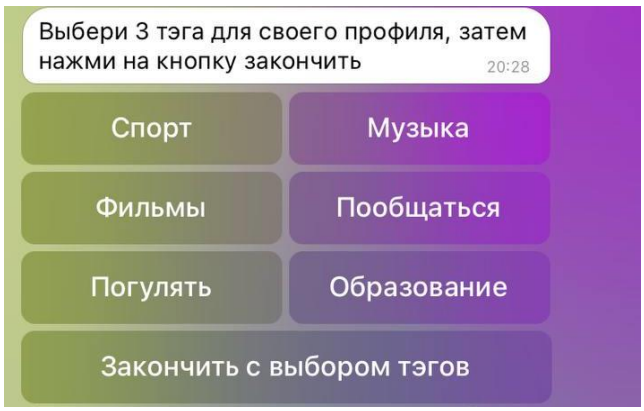


Рисунок 9 – Выбор до 3 тэгов пользователем

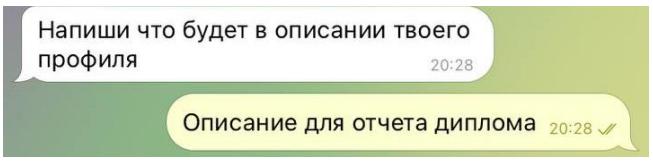


Рисунок 10 – Ввод описания для профиля

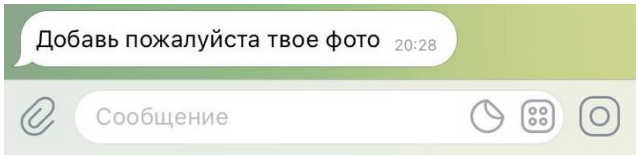


Рисунок 11 – Добавление фото пользователем

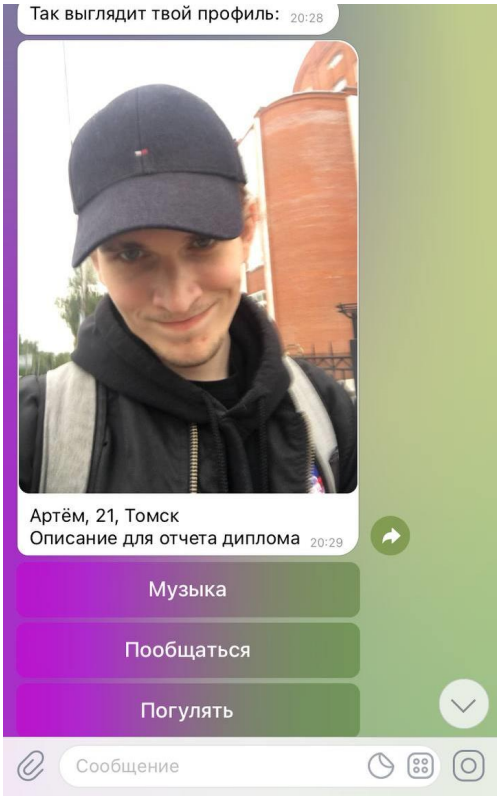


Рисунок 12 – Профиль пользователя

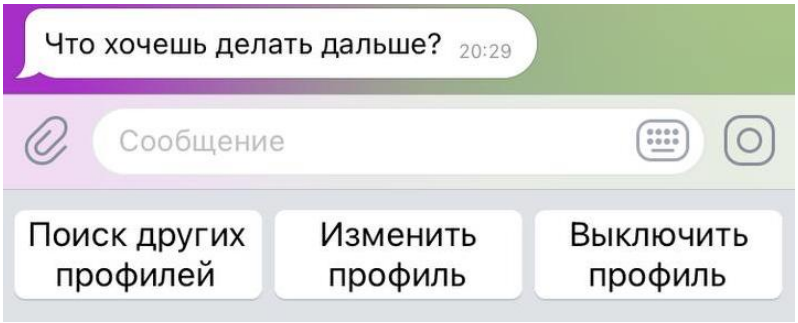


Рисунок 13 – Главное меню бота

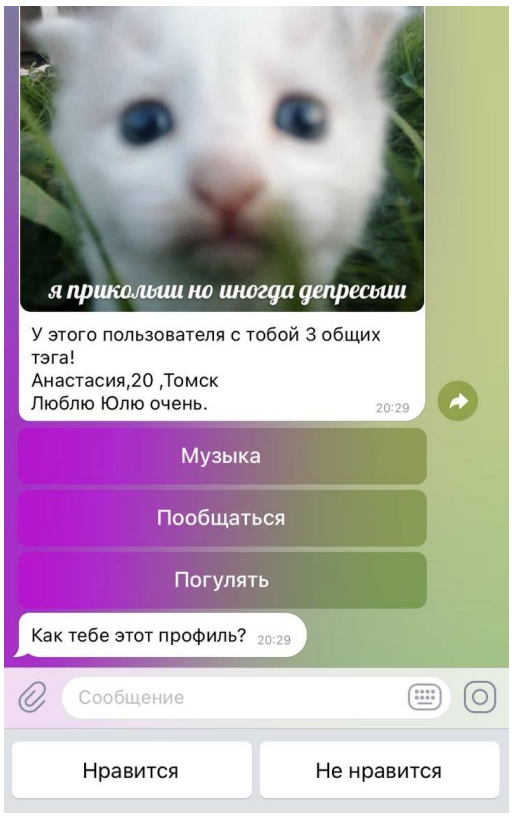


Рисунок 14 – Выдача пользователя в поиске

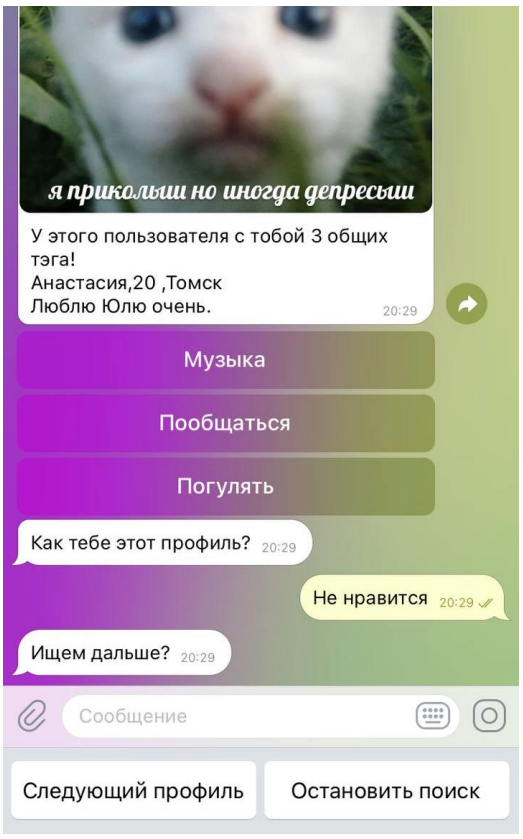


Рисунок 15 – Оценка профиля и продолжение/остановка поиска

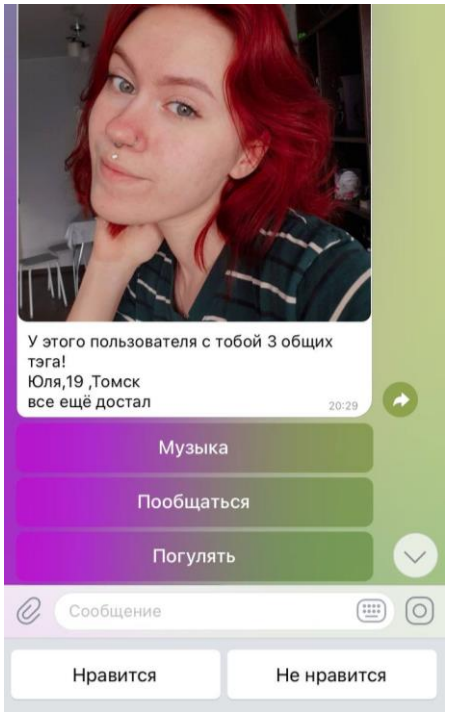


Рисунок 16 – Следующий в очереди профиль

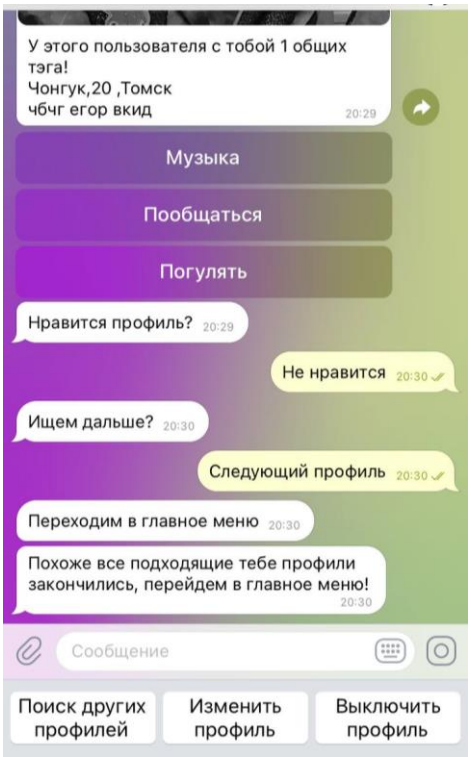
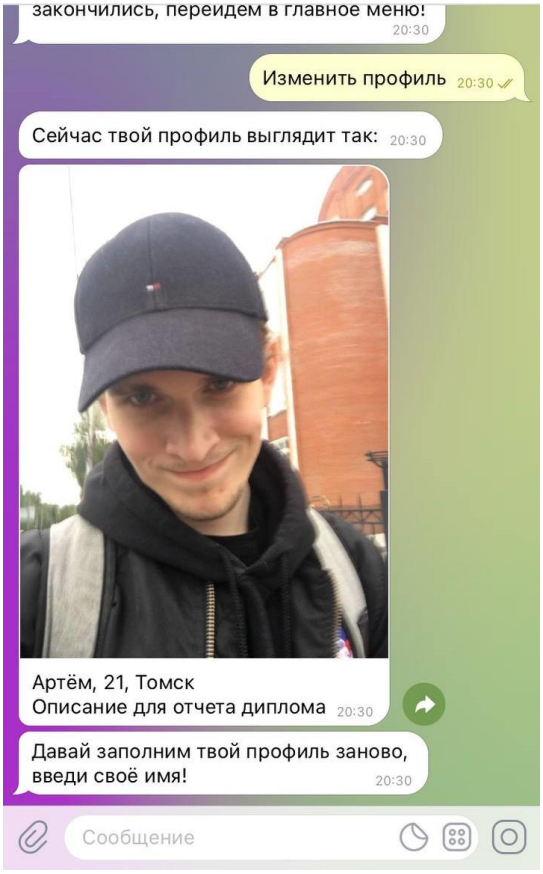


Рисунок 17 – Остановка поиска и переход в главное меню



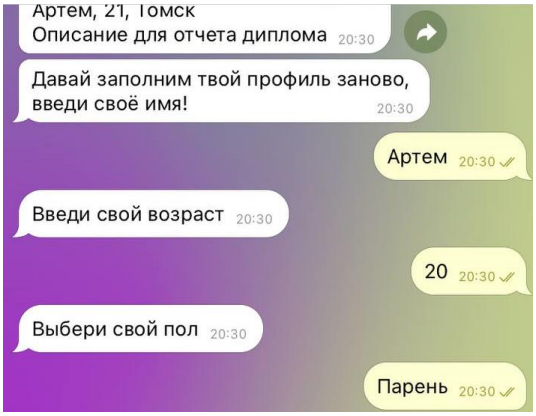


Рисунок 18 – Изменение профиля

Выключение профиля

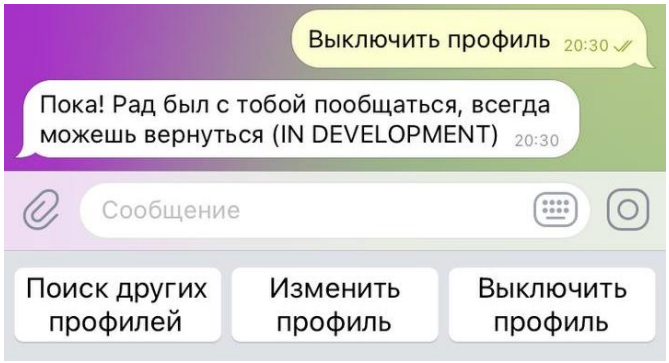


Рисунок 19 – Выключение профиля и прекращение работы с ботом

1.2. Тэговая система рекомендаций

В проектируемом чат-боте для отбора профилей кроме регистрационных данных (возраст, город, пол) реализована возможность использования дополнительных дескриптовов - тэгов.

Тэги — это дескрипторы для дополнительного описания интересов пользователя. Пользователю предлагается выбрать до трех тэгов.

В дальнейшем, при отборе профилей, сравниваются тэги двух пользователей. Определяется количество пересечение тэгов у пользователей и на основании этого формируется очередь пользователей, которая предъявляется чат-ботом участникам коммуникации. С увеличением количества общих тэгов вероятность коммуникации увеличивается, т.к. такие пользователи получают приоритет при формировании очереди. На данный момент пользователям предлагается возможность выбрать до трех тэгов из шести (рисунок 20). Таким образом, учитывая число сочетаний $C^3_6 = 20$, длина очереди теоретически достигает 37 млн × 20.

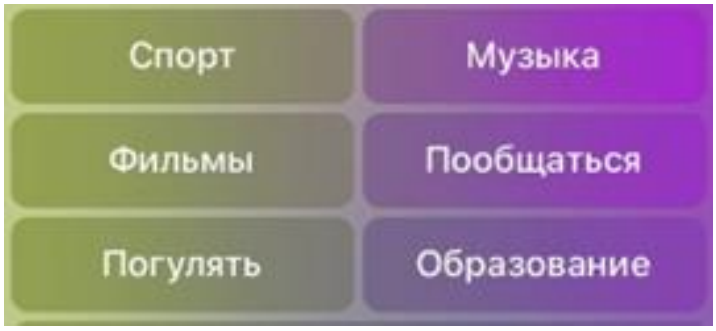


Рисунок 20 – Примеры тэгов для выбора

При выборе тэгов возможны следующие ситуации, проиллюстрированные на рисунках ниже:

Рисунок 21 - Один общий тэг у 2 пользователей

Рисунок 22 – Два общих тэга у 2 пользователей

Рисунок 23 – У пользователей нет общих тэгов

2. Разработка базы данных

2.1. Концептуальное проектирование

Атрибуты сущности пользователя, который взаимодействует с ботом:

- TelegramID (Первичный ключ) – уникальный идентификатор пользователя Telegram
- Name – Имя, вводимое пользователем
- Age - Возраст, вводимый пользователем
- Gender – Пол, выбираемый пользователем
- City – Город, вводимый пользователем
- Description – Описание, вводимое пользователем
- SearchGender – Пол поиска людей, выбираемый пользователем
- Tag1 - Тэг 1, выбранный пользователем
- Tag2 - Тэг 2, выбранный пользователем
- Tag3 - Тэг 3, выбранный пользователем
- ActiveStatus - Статус активности аккаунта, устанавливаемый в начале работы с ботом пользователю, затем изменяемый при остановке бота
- TelegramUsername – Ник пользователя в Telegram
- Photo – название фотографии, которую загружает пользователь
- LastmatchID – уникальный последний идентификатор Telegram, профиль пользователя который понравился пользователю

Приведем краткое описание атрибутов:

- TelegramID – первичный ключ для сущности пользователя, является целыми числом (int). По нему осуществляется поиск, вставка, изменение и удаление информации о пользователях бота.
- Имя – псевдоним пользователем, который будет отображаться в его профиле, является строкой (string).
- Возраст – указанный пользователем возраст, который будет отображаться в его профиле, является целым числом (int).

- Пол – пол пользователя с возможными значениям (М, Ж) , который будет отображаться в его профиле, является строкой единичного размера (string).
- Город – представляет из себя набранный пользователем город, который будет отображаться в его профиле, является строкой (string).
- Описание – представляет из себя набранное пользователем описанием его профиля, которое будет отображаться в его профиле, является строкой (string).
- Пол поиска людей – представляет из себя набранный пользователем пол с возможными значениям (М, Ж, ALL) , по которому будет происходить поиска профилей, является строкой единичного размера (string).
- Тэги – представляют из себя выбранные пользователем интересы, которые будет отображаться в его профиле, являются строками (string).
- Статус активности – представляет из себя логическую переменную типа bool, которая по умолчанию ставится пользователю как true. Определяет все ещё ли пользователь пользуется ботом. Если прекратил – ставится false.
- TelegramUsername – представляет из себя никнейм пользователь в Telegram. Является переменной типа string. Позволяет предоставлять данные о пользователях при взаимном согласии.
- Фото – представляет из себя переменную типа string. Является названием фото, которое пользователь загрузил боту.
- LastmatchID – представляет из себя переменную типа int. Является TelegramID для последнего пользователя, профиль которого понравился другому пользователю.

2.2. Построение даталогической модели

Построим даталогическую модель базы данных:

Таблица 2 – Даталогическая модель базы данных. Таблица *users*

Атрибут	Тип	Размер, байт	Допустимые значения	Значения по умолчанию	Ключи
TelegramID	Целочисленный	4	Уникальный	-	ПК: TelegramID
Name	Символьный	200	-	-	
Age	Целочисленный	4	-	-	
Gender	Символьный	1	-	-	
City	Символьный	50	-	-	
Description	Символьный	500	-	-	
SearchGender	Символьный	3	-	-	
Tag1	Символьный	20	-	-	
Tag2	Символьный	20	-	-	
Tag3	Символьный	20	-	-	
ActiveStatus	Логический	1	True/False	True	
TelegramUsername	Символьный	100	-	-	
Photo	Символьный	100	-	-	
LastmatchID	Целочисленный	4	-	-	

2.3. Реализация базы данных

Создание таблицы:

```
CREATE TABLE IF NOT EXISTS public.users
(
    telegramid integer NOT NULL,
    name character varying(200) COLLATE pg_catalog."default",
    age integer,
    gender character varying(1) COLLATE pg_catalog."default",
    city character varying(50) COLLATE pg_catalog."default",
    description character varying(500) COLLATE pg_catalog."default",
    searchgender character varying(50) COLLATE pg_catalog."default",
    tag1 character varying(20) COLLATE pg_catalog."default",
    tag2 character varying(20) COLLATE pg_catalog."default",
    tag3 character varying(20) COLLATE pg_catalog."default",
    activestatus boolean DEFAULT true,
    telegramusername character varying(100) COLLATE pg_catalog."default",
    photo character varying(100) COLLATE pg_catalog."default",
    lastmatchid integer,
    CONSTRAINT users_pkey PRIMARY KEY (telegramid)
)
```

2.4. Выбор СУБД

При выборе СУБД мы рассмотрели Oracle SQL, PostgreSQL, MySQL. Выбор был сделан в пользу PostgreSQL, потому что данная СУБД предоставляет:

- Возможность масштабирования базы при большом количестве пользователей
- Для Python есть хорошо задокументированная библиотека psycopg2, которая позволяет взаимодействовать с PostgreSQL.

	telegramid [PK] integer	name character varying (…	age integer	gend chara	city character va	description character varying (500)	searchg charact
1	287660072	Артём	21	М	Томск	Хочу сосиску	All
2	696357154	Александра	20	Ж	Томск	Кто не любит Даби-…	All
3	1141136455	Чонгук	20	Ж	Томск	чбчг егор вкид	All
4	1763610907	Alex	223	М	Томск	Хочу богатую женщ…	M
5	341563240	Машуня	223	Ж	Томск	куку ляля	M
6	753357258	Катя	20	Ж	Томск	любите юлю	All
7	983108521	Юля	19	Ж	Томск	все ещё достал	All
8	941438389	Анастасия	20	Ж	Томск	Люблю Юлю очень.	All

Рисунок 24 – Пользователи в БД

tag1 character varyin	tag2 character vary	tag3 character varyi	actives boolea	telegramusern character vary	photo character varying (100)	lastmatchid integer
Музыка	Пообщаться	Погулять	true	Vislysya	AgACAgIAAxkBAAITn2Kg1PNE…	983108521
Музыка	Пообщаться	[null]	true	ourmint3	AgACAgIAAxkBAAIXvGKklaq1_…	983108521
Музыка	Спорт	Фильмы	true	yagamikk	AgACAgIAAxkBAAIXW2KkkyFii…	983108521
Музыка	Пообщаться	Пиво	true	Alexanax357	AgACAgIAAxkBAAIThGKg1Fu…	[null]
Пообщаться	Образован…	Музыка	true	maffonka	AgACAgIAAxkBAAITwGKg2Gr2…	287660072
Музыка	Погулять	Пообщаться	true	itskkaty	AgACAgIAAxkBAAIWm2Kki_jw…	287660072
Музыка	Погулять	Пообщаться	true	bomgosh	AgACAgIAAxkBAAITD2KgzlIRC…	287660072
Музыка	Пообщаться	Погулять	true	anvi_17	AgACAgIAAxkBAAIYVGKkml3o…	[null]

Рисунок 25 – Пользователи в БД

3. Разработка алгоритма и программы

3.1. Выбранные технологии разработки

Для программной реализации социального чат-бота выбран язык Python. Выбран он был из-за нижеприведенных особенностей:

- Python имеет много методов для обработки текста, что позволяет быстро и качественно работать с ним.
- Python имеет большое количество библиотек с разным функционалом. В данной работе была как раз использована библиотека pytelegrambot. Pytelegrambot – api для создания бота.
- Python является очень востребованным языком, что означает он хорошо задокументирован, а также существует много различных учебных материалов по его освоению.
- Также в сравнении с другими языками программирования Python имеет динамическую типизацию. Это означает то, что в нем возможно определять переменные в момент присваивания им значений, а не в моменте их инициализации.

pyTelegramBotAPI

Эта библиотека позволяет реализовывать синхронный функционал Telegram ботов. В ней есть методы создания логики бота, его пользовательского интерфейса.

Psycopg2

Psycopg2 – одна из самых популярных адаптаций взаимодействия с PostgreSQL для языка программирования Python. Она обладает широким функционалом и большой скоростью выполнения команд. В ней присутствует возможность осуществления асинхронной и синхронной коммуникации, курсоров, позволяющих обрабатывать данные.

3.2. Реализация и описание алгоритмов

- Общие тэги и составление очереди людей, подходящих под критерии возраста, города и тэгов

```
with connection.cursor() as cursor:
    cursor.execute(
        "SELECT name, age, gender, city, description, searchgender, tag1, tag2, tag3, activestatus, "
        "telegramusername, telegramid, photo FROM users WHERE telegramid = %s;", (message.from_user.id,) )
    myprofiledata = cursor.fetchone()##my profile
```

Рисунок 26 – Получение своих данных для взаимодействия с поиском

```
mytagset = set([myprofiledata[6],myprofiledata[7],myprofiledata[8]])
mytags = list(mytagset)
for tag in mytags:
    if tag == None or tag == "None":
        mytags.remove(tag)
mytags = set(mytags)
```

Рисунок 27 – Формирование непустых тэгов

```
cursor.execute(
    "SELECT telegramid, name, age, gender, city, description, searchgender, tag1, tag2, tag3, activestatus, telegramusername, photo FROM users"
    " EXCEPT(SELECT telegramid, name, age, gender, city, description, searchgender, tag1, "
    "tag2, tag3, activestatus, telegramusername, photo FROM users WHERE telegramid = %s)", (message.from_user.id,)
)
query3tags=[]
query2tags=[]
query1tag=[]
queryall=[]
uselessquerry=[]
sharedtags = []
allQuerry = cursor.fetchall()
```

Рисунок 28 - Формирование очередей, удовлетворяющих критериям и общим тэгам

```
personTagSet=set([person[7],person[8],person[9]])
persontags = list(personTagSet)
for tag in persontags:
    if tag == None or tag == "None":
        persontags.remove(tag)
personTagSet = set(persontags)
sharedtagsSet = mytagset.intersection(personTagSet)
sharedtags = list(sharedtagsSet)
```

Рисунок 29 – Обработка непустых тэгов найденных профилей

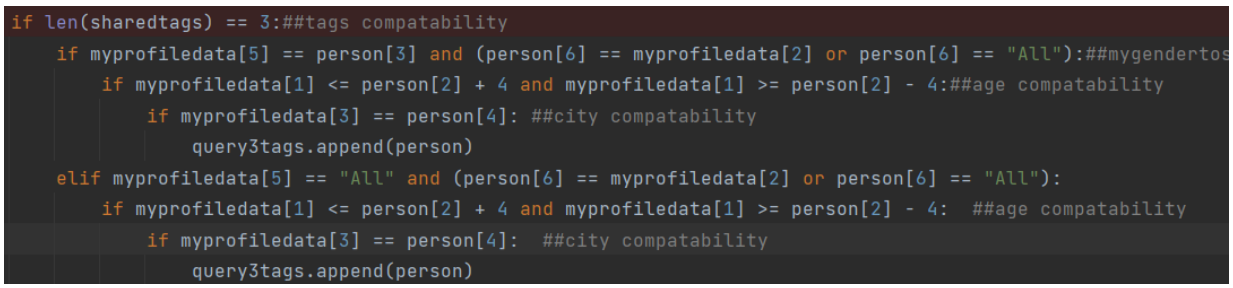


Рисунок 30 – Критерии возраста и города в зависимости от количества тэгов

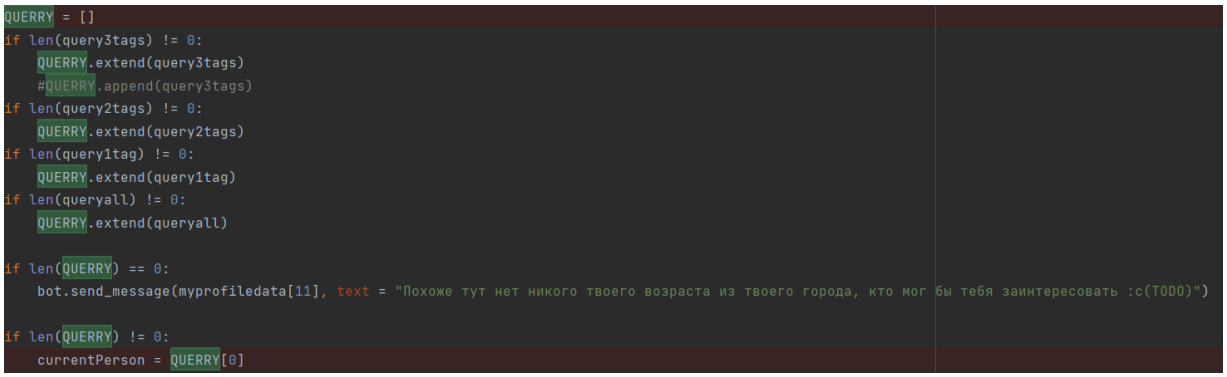


Рисунок 31 – Добавление всех подходящих профилей к общей очереди

- Поиск пользователей из очереди, запрос на общение

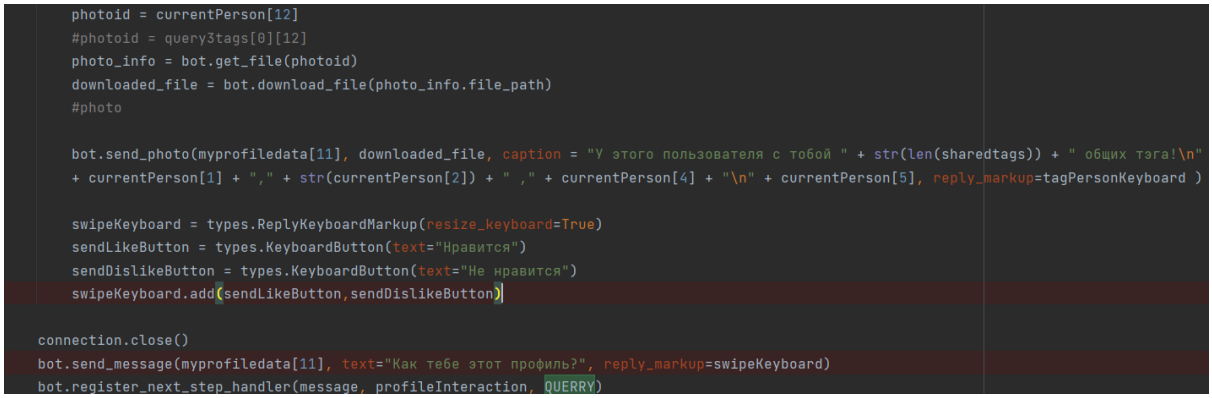


Рисунок 32 – Получение рекомендованного профиля и клавиатура взаимодействия

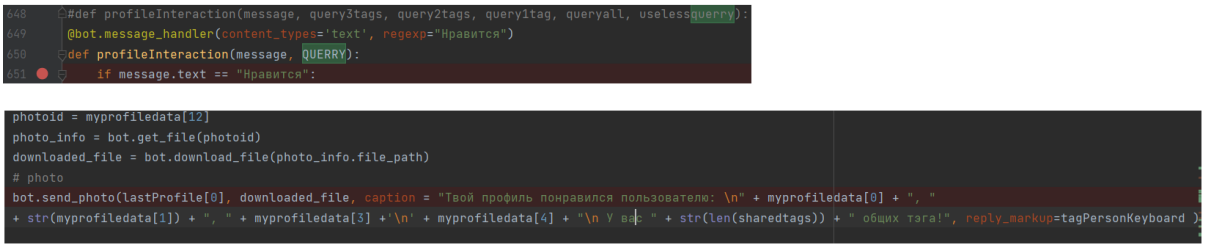


Рисунок 33 – Взаимодействие с пользователем, чей профиль понравился другому пользователю

```
@bot.message_handler(content_types='text', regexp="Лайк" )
def answerRequest(message):
bot.send_message(data[0], text= "Приятно вам пообщаться\n" + "t.me/" + mydata[2])
bot.send_message(mydata[0], text= "Приятно вам пообщаться\n" + "t.me/" + data[2])
```

Рисунок 34 – Реализация обмена контактами пользователей при взаимном согласии

3.4. Регистрация пользователя

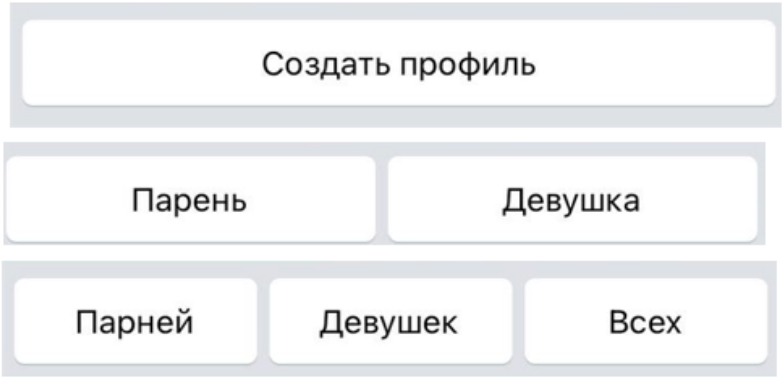


Рисунок 35 – Клавиатуры взаимодействия для регистрации пользователя

3.5. Главное меню

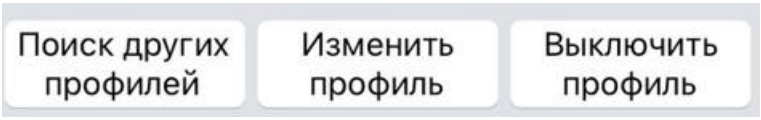


Рисунок 36 – Клавиатура главного меню

```
mainkeyboard = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
button_searchProfiles = types.KeyboardButton(text="Поиск других профилей")
button_changeProfile = types.KeyboardButton(text="Изменить профиль")
button_quitProfile = types.KeyboardButton(text="Выключить профиль")
mainkeyboard.add(button_searchProfiles, button_changeProfile, button_quitProfile)
```

Рисунок 37 – Реализация клавиатуры главного меню

3.6. Поиск профилей других пользователей

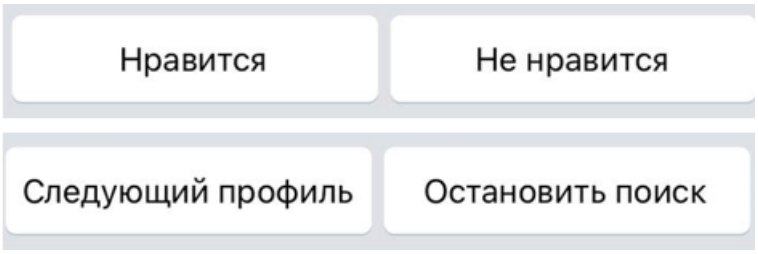


Рисунок 38 – Клавиатуры взаимодействий с поиском профилей

```
swipeKeyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
sendLikeButton = types.KeyboardButton(text="Нравится")
sendDislikeButton = types.KeyboardButton(text="Не нравится")
swipeKeyboard.add(sendLikeButton,sendDislikeButton)
```

Рисунок 39 – Реализация клавиатуры просмотра профиля

3.7. Изменение своего профиля

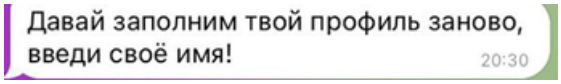


Рисунок 40 – Сценарий изменения заполнения профиля

3.8. Выключение профиля

Рисунок 41 – Сценарий выключения профиля и окончания работы с ботом

4. Задание для раздела «Концепция стартап-проекта»

Студенту:

Группа	ФИО
0В8Б	Брылин Артем Владимирович

Школа	**ИЯТШ**	**Направление**	01.03.02 Прикладная математика и информатика
Уровень образования	бакалавриат		

Перечень вопросов, подлежащих разработке:	
Проблема конечного потребителя, которую решает продукт, который создается в результате выполнения НИОКР (функциональное назначение, основные потребительские качества)	Продукт позволяет его потребителям взаимодействовать с другими людьми, тем самым социализируя пользователей.
Способы защиты интеллектуальной собственности	Регистрация товарных знаков и защита авторских прав на программу ЭВМ и базу данных
Объем и емкость рынка	Объем российского рынка составляет от 70 млн $.
Современное состояние и перспективы отрасли, к которой принадлежит представленный в ВКР продукт	Рынок активно развивается. Количество потребителей постоянно возрастает.
Себестоимость продукта	Оценена в 299000 рублей
Конкурентные преимущества создаваемого продукта	Удобный интерфейс, понятный дизайн, популярная используемая платформа продвижения, более персонализированные настройки пользователя.
Сравнение технико-экономических характеристик продукта с отечественными и мировыми аналогами	Наличие тэговой системы и геймификация элементов графического пользовательского интерфейса
Целевые сегменты потребителей создаваемого продукта	Мужчины и женщины в возрасте от 16 до 45 лет.
Бизнес-модель проекта	Монетизация проекта будет производиться путем рекламы.

Каналы продвижения продукта	Интернет, социальные сети, мессенджеры
Перечень графического материала:	
При необходимости представить эскизные графические материалы (например, бизнес-модель)	*Бизнес – модель Остервальдера - Пинье*

Дата выдачи задания для раздела по линейному графику	

Задание выдал консультант по разделу «Концепция стартап-проекта» (со-руководитель ВКР):

Должность	**ФИО**	**Ученая степень, звание**	**Подпись**	**Дата**
Доцент	Селевич Ольга Семеновна	к.э.н.		

Задание принял к исполнению студент:

Группа	**ФИО**	**Подпись**	**Дата**
0В8Б	**Брылин Артем Владимирович**		

4.1. Проблема конечного потребителя

В современном мире становится все больше людей, которые предпочитают знакомиться в интернете, нежели чем в реальной жизни. Это связано с различными причинами, которые привели общество к тому, что многие люди перестали знакомиться и общаться в реальном мире как раньше.

Первая и, пожалуй, одна из самых весомых — это неуверенность в себе. Многие люди страдают от комплексов. Но внешние недостатки, кажущиеся чем-то отталкивающим, легко можно скрыть при общении в социальных сетях. В отличие от знакомств на улице или в общественных местах, интернет позволяет создать любой образ, слегка изменив некоторые факты.

Также большинство людей недовольны жизнью, которую проживают. Они сбегают в социальные сети, чтобы найти там «убежище». Подростки, которые не могут наладить общий язык с родителями; супруги, уставшие от быта; люди, переживающие непростой жизненный период, — всем им нужен тот, кто по ту сторону экрана просто выслушает или даст совет. Никаких обязательств, никакой ответственности.

Кроме того, многие считают, что окружающие их люди не соответствуют тем требованиям, которые они предъявляют. Например, офисный работник, который общается в основном с коллегами, но при этом ищет девушку-модель. Или студентка, считающая однокурсников слишком бедными для того, чтобы строить с ними отношения. Таким людям, конечно же, проще найти человека в интернете. Достаточно ввести нужные параметры и выбрать из огромного множества представленных вариантов.

И наконец, главная причина общения онлайн — банальная нехватка времени. Огромные объёмы работы, вечная спешка, ворох дел и ещё немаленький список того, что нужно делать каждый день. В результате человек крутится как белка в колесе и не замечает ни прохожих, ни заинтересованных взглядов. Те люди, которые вовремя осознали этот тренд – начали создавать приложения и сайт, тем самым формируя рынок сферы онлайн знакомств.

Соответственно, можно описать проблему конечного потребителя как «приложить минимальные усилия, чтобы познакомиться с другими людьми в интернете».

4.2. Способы защиты интеллектуальной собственности

Главный способ для защиты интеллектуальной собственности разрабатываемого решения – получение патентов на отдельные элементы, из которых оно состоит. Среди них:

- Название бота
- Логотип бота
- Дизайн интерфейса
- Программный код

Свою известность и популярность среди пользователей мобильные приложения получают благодаря своему название и логотипу. Для их защиты следует зарегистрировать товарный знак для них. Также, для мобильных приложения наличие товарного знака может являться необходимым для дистрибуции в платформах, где осуществляется продажа/загрузка приложений.

Также, чтобы защитить программу для ЭВМ и Базу данных, используемых в боте, следует зарегистрировать их в Роспатенте. В соответствии с 1259 с. ГК РФ программа для ЭВМ является объектом авторских прав. Регистрация вышеперечисленного позволит получить свидетельство, при котором автор будет являться правообладателем с исключительным правом на разработку. Также это позволит в упрощенном виде осуществлять передачу авторских прав, либо их продажу. При судебном разбирательстве это свидетельство будет доказательством исключительных и авторских прав на программу ЭВМ и Базу данных, используемую в боте.

4.3. Объем и емкость рынка

Объем рынка онлайн-знакомств является достаточно большим, а также активно развивается. Объем рынка к концу 2020 года составляет примерно $3 млрд, а рост количества пользователей составляет 7,5%. Глобальное количество

пользователей приложений для знакомств составляет более 300 миллионов человек. Прогнозируется достижение $8 млрд объема рынка к 2024 году.

По некоторым данным рынок в России оценивается в $92 млн[30].

4.4. Современное состояние и перспективы отрасли, к которой принадлежит представленный в ВКР продукт

В данное время отрасль активно развивается, количество пользователей постоянно и объем рынка постоянно растет. Отрасль является перспективной из-за того, что знакомство в интернете проходит быстрее и легче, чем оно же в реальной жизни.

По разным оценкам в данный момент существует около 8 тысяч приложений для знакомств, а также постоянно запускаются новые сервисы.

Одним из факторов бурного развития этой сферы является её эффективность. Помимо упрощения приложений для более удобного взаимодействия с пользователями, некоторые компании улучшают свои поисковые алгоритмы. Для этого привлекается машинное обучение и используется искусственный интеллект. Это позволяет пользователям находить более подходящих для них людей. Помимо этого, текущее состояние отрасли использует геймификацию в процессе работы с пользователем. Большое количество разнообразных анимированных элементов и интересных дизайнерских решений для интерфейсов и взаимодействия с пользователями. Такой подход делает процесс взаимодействия пользователей с приложениями более запоминающимся, тем самым давая им позитивные эмоции, которые выражаются в том, что люди могут пользоваться приложениями достаточно длительное время, а также возвращаться в них через какое-либо время отсутствия.

4.5. Себестоимость продукта

Разработка продуктов данной тематики в среднем обходится от $25000 до $150000, если разрабатывается полноценное приложение, где команда разработчиков составляет 5-10 человек.

В случае с разработкой бота внутри мессенджера тратится значительно меньшее количество средств, так как необходимость полностью реализовывать авторизацию, выявление геолокации пользователя и проектирование графических пользовательских интерфейсов отсутствует.

На разрабатываемый продукт было потрачено приблизительно 200 часов активной разработки(программирования). На доведение алгоритмов от стадии идеи до работающего прототипа – 20 часов. На проектирование графического интерфейса – 10 часов.

Проектом занимался 1 разработчик.

Если брать среднюю стоимость рабочего часа для одного программиста, который занимается реализацией бота, можно взять условную стоимость 1 часа работы как 1000 рублей.

Соответственно, конечная стоимость этапов разработки:

- Написание кода – 200 ч. × 1000 руб. = 200 000 руб.
- Разработка алгоритмов - 20 ч. × 1000 руб. = 20 000 руб.
- Реализация графического интерфейса 10 ч. × 1000 руб. = 10 000 руб.

Суммарная стоимость разработки оценивается в размере 230 тысяч рублей, при затрате 230 часов работы. Добавляем 30% для страховых взносов. Учитываем, что стоимость необходимого программного обеспечения для разработки отсутствует (т.е. бесплатно), а интернет и компьютерная техника уже обеспечены.

Итого – 299000 рублей.

4.6. Конкурентные преимущества создаваемого продукта

У данного продукта будут конкурентные преимущества в виде удобного и понятного пользовательского интерфейса, также будут расширенные рекомендации по поиску других людей. Также можно отметить тот факт, что приложения мессенджеры являются гораздо более удобными в пользовании по отношению к социальным сетям. В качестве платформы для создания выбран

бесплатный кроссплатформенный мессенджер мгновенного обмена сообщений – Telegram.

Основной особенностью разрабатываемого бота будет наличие тэгов, как интерактивного элемента профилей пользователя. У схожих ботов отсутствует тэговая система, реализованная в полноценных приложениях. Ниже представлена реализация выбора тэгов в боте.

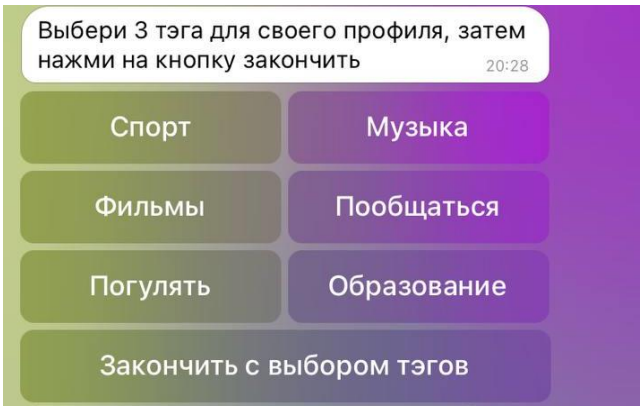


Рисунок 42 – Тэги пользователя

Пользователь может иметь до 3 тэгов на выбор, а также в дальнейшем изменять их. В зависимости от выбранных тэгов для него будет формироваться список профилей других пользователей с общими тэгами. Если общих тэгов не будет, но пользователь будет устраивать первого пользователя по другим критериям – возрасту и городу, то тогда он тоже может попасть в этот список, но будет отображаться позже, чем более подходящие профили.

На рисунке ниже изображен профиль одного пользователя.

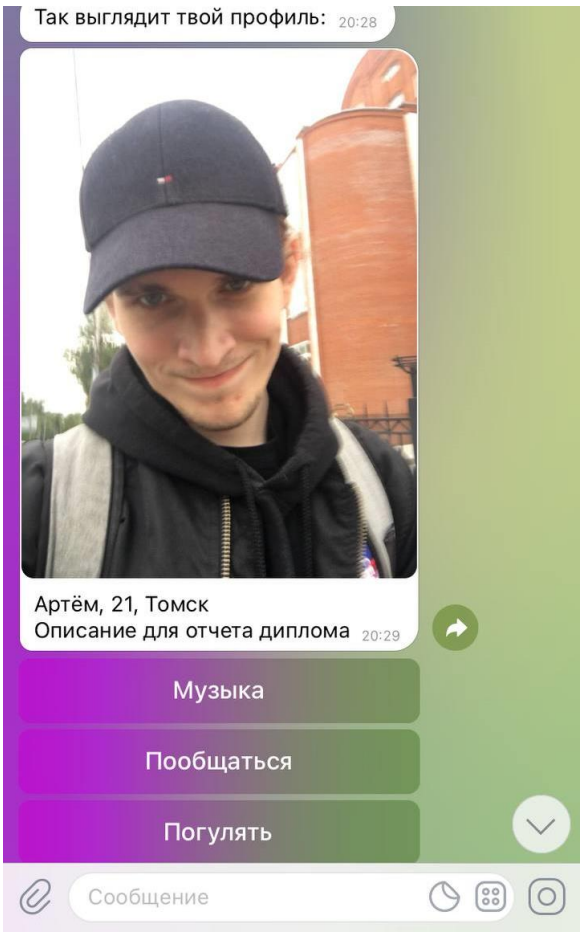


Рисунок 43 – Профиль пользователя

Как видно, тэги являются выделяющимися элементами профиля, даже при отсутствии добавления в них emoji, которые являются элементами геймификации, при использовании их в боте. При необходимости можно разнообразить профили с помощью них, а также сделать интуитивно понятные иконки для основных активностей – например для спорта можно поставить emoji футбольного мяча, для музыки – emoji музыкальной ноты.

Помимо вышеперечисленного отличия от конкурентов, существуют также пользовательские клавиатуры для взаимодействия с ботом. Пример приведен на рисунке ниже.

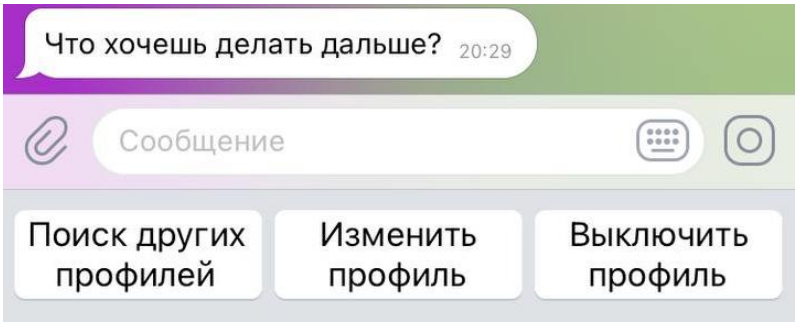


Рисунок 44 – Клавиатуры взаимодействия с ботом

Аналогичные элементы геймификации и взаимодействия с ботом присутствуют у конкурентов, но можно по желании добавить выделяющихся элементов в их дизайне (те же emoji), а также реализовать функционал, который может получать какой-то популярный профиль по нажатию. Это также может являться уникальным отличием от конкурентов.

Бот был реализован в мессенджере, а не в социальных сетях. Это связано с тем, что сейчас функционал различных мессенджеров уже ничем не уступает социальным сетям. И как правило их интерфейс и функционал – удобнее чем у социальных сетей. Именно поэтому было решено разрабатывать бота в Telegram.

4.7. Сравнение технико-экономических характеристик продукта с отечественными и мировыми аналогами

В качестве сравнения бота с существующими аналогами приложений и ботов для онлайн знакомств возьмём уже существующие успешные продукты, такие, как приложение для знакомств “Tinder”, “Дай Винчик” - бот для знакомств в мессенджере Telegram.

Таблица 3 – Сравнение продуктов-аналогов

Пункт сравнения	Разрабатываемое решение	Tinder	Дай Винчик
Модель распространения	Бесплатный, коммерциализируется рекламой	Бесплатный, коммерциализируется необязательной платной	Бесплатный, коммерциализируется рекламой

		подпиской (Freemium)	
Целевая аудитория	В основном молодая: 16-30 лет	В основном взрослая: 18-40	В основном молодая: 16-30 лет
Профиль пользователя	В профиле есть фотографии/видео, тэги, описание, город, возраст. Все заполняется пользователем	В профиле есть фотографии/видео, геолокация, город, возраст, описание, тэги. Все заполняется пользователем.	В профиле есть фотографии/видео, описание, город, возраст. Все заполняется пользователем
Платный функционал	Отсутствует	Дополнительные платные возможности в виде статуса онлайна, настройки персонализации	Отсутствует
Стоимость разработки	299000 рублей	$25-30 тыс.	500000 рублей

Исходя из вышестоящей таблицы уместно будет сравнить отличия бота Дай Винчик от разрабатываемого решения.

Таблица 4 – Сравнение ближайшего аналога с разрабатываемым решением

Пункт сравнения	Разрабатываемое решение	Дай Винчик
Особенности профиля пользователя	В профиле есть тэговая система	Тэговая система отсутствует
Дизайн графического интерфейса	Большее количество кнопок как элементов	Дизайн бота давно не изменялся, количество

	настройки профиля и взаимодействия с ботом. Обновление дизайна	кнопок как элементов настройки профиля и взаимодействия с ботом меньше
Количество пользователей	Количество пользователей при выпуске бота будет меньше, чем у тех продуктов, кто долго находится на рынке	Большое количество пользователей. Основные активные пользователи слишком часто видят повторяющихся людей

Исходя из вышестоящей таблицы можно сделать вывод что качество разрабатываемого бота будет более высоким, чем у его ближайшего аналога. Из минусов можно выделить сильную разницу в количестве пользователей, которую можно нивелировать свежестью продукта.

4.8. Целевые сегменты потребителей создаваемого продукта

Так как разрабатываемый бот является продуктом для социального взаимодействия, а именно онлайн знакомств, то для него можно выделить основные сегменты потребителей, в зависимости от их нужд.

Таблица 5 - Сегменты потребителей

Сегмент потребителей	Нужды потребителей этого сегмента
Люди, испытывающие проблемы с социализацией в реальном мире. Как правило это люди возрастов от 16 до 25 лет	Пообщаться, завести друзей, найти компанию по общим интересам. Найти себе какое-то хобби или занятие в реальной жизни
Люди, которые хотят анонимно обсудить что-то с неизвестным человеком.	Пообщаться, возможно завести друга или товарища по переписке

Люди любых возрастов, как правило молодые.	
Люди, которым нужны собеседники для социологии или своих нужд. Люди любых возрастов.	Пообщаться с собеседником, который имеет свое мнение/разбирается в каком-то вопросе
Одинокие люди, которые хотят найти себе романтические отношения. Люди любых возрастов, как правило молодые.	Пообщаться и найти человека для свиданий и романтического общений
Люди, которые хотят весело провести время Люди любых возрастов	Найти какое-либо занятие или сходить на прогулку/в какое-то место с новыми людьми

Исходя из вышеприведенной таблицы можно сделать вывод, что у потребляемого продукта в лице разрабатываемого бота есть большой потенциал к развитию, из-за наличия разных потребительских сегментов, где у каждого сегмента есть свои нужды.

4.9. Бизнес-модель проекта

Бизнес–модель проекта по А. Остервальдеру

<table>
<tr>
<td rowspan="2">Ключевые партнеры
Различные интернет-сообщества</td>
<td>Ключевые виды деятельности
Обеспечение пользователей профилями других людей</td>
<td rowspan="2">Ценностные предложения
Пользователь получает контакты других пользователей</td>
<td>Взаимоотношения с клиентами
Бот не требует затрат, легко осваиваемый</td>
<td rowspan="2">Потребительские сегменты
Люди, испытывающие проблемы с социализацией
Люди, ищущие общения
Люди, которые хотят активно провести время с новыми людьми</td>
</tr>
<tr>
<td>Ключевые ресурсы
Telegram
База данных</td>
<td>Каналы сбыта
Интернет
Социальные сети
Мессенджеры</td>
</tr>
<tr>
<td colspan="2">Структура издержек
Затраты на обеспечение работы бота и поддержку инфраструктуры
Затраты на разработку бота</td>
<td colspan="3">Потоки поступления доходов
Монетизация бота с помощью внутренней рекламы</td>
</tr>
</table>

Рисунок 45 - Бизнес модель проекта

На рисунке выше представлена бизнес-модель проекта рекомендательного бота для социального взаимодействия.

Ключевые партнеры

В качестве ключевых партнеров были выделены интернет-сообщества, взаимодействие с которыми может помочь развить бота путем привлечения целевой аудитории. Это поможет с решением основной трудности проекта – наличием пользователей.

Виды деятельности

Главным видом деятельности этого проекта является возможность пользователем поиска других людей по определенным критериям.

Ценностное предложение

Ценность предложения представляет из себя большое количество пользователей, которое может осуществлять социальное взаимодействие между друг другом.

Отношение с покупателями

Пользователем будет легко взаимодействовать с продуктом, так как его архитектура и функционал очень просты в освоении. Помимо этого пользователю не нужно ничего платить, чтоб воспользоваться реализацией проекта.

Сегментация потребителей

Сегменты потребителей уже были описаны ранее. Бот будет предоставлять качественное решение их проблем и реализацию их желаний.

Структура затрат

Проект не является особо затратным. Из необходимого для поддержания его работы нужно учесть хранение пользовательских данных на каком-либо хостинге. Также есть затраты на разработку самого бота, а если его функционал нужно будет обновить – то это не будет масштабной тратой.

Ключевые ресурсы

В качестве ключевых ресурсов для проекта нужно иметь Telegram и базу данных, которые позволят взаимодействовать пользователям с ботом, и в дальнейшем между собой.

Каналы взаимодействия

Под каналами взаимодействия можно рассматривать интернет, социальные сети и мессенджеры. Можно обеспечить взаимодействия с другими тематическими сообществами.

Поток поступления доходов

Бот будет бесплатным для использования пользователями. В нем не будет присутствовать платная подписка, а монетизация будет осуществляться с помощью рекламы других площадок, интернет-сообществ и других ресурсов.

4.10. Каналы продвижения продукта

В качестве продвижения разрабатываемого продукта следует использовать интернет, социальные сети и мессенджеры. Продвижение осуществляется рекламой бота и перенаправлением пользователей, увидевших рекламу, по ссылке в Telegram бота. Самым эффективным способом продвижения можно считать социальную сеть Вконтакте. Она имеет схожую целевую аудиторию с

разрабатываемым продуктом, что позволит привести большее количество клиентов в бота. В социальной сети Вконтакте стоимость продвижения может начинаться от 500 рублей. Вконтакте рекомендует цену за 1000 просмотров рекламной записи в размере 15-40 рублей. Но для начала следует поставить 1 рубль за 1000 просмотров, и в зависимости от статистики продвижения уже повышать её.

Также существуют альтернативные возможности продвижения, но выгода от них будет меньше, чем от предложенного способа. Среди них – реклама в каналах самого Telegram. Стоимость продвижения и потенциальное количество пользователей зависит от популярности каналов. Можно ориентироваться на каналы до 5000 подписчиков, в которых 1000 просмотров можно оценивать в размере 300-500 рублей.

5. Социальная ответственность

5.1. Введение

Областью применения проекта разработки в работе является информационно-телекоммуникационная сеть Интернет, в частности свободный кроссплатформенный сервис мгновенной отправки сообщений мессенджере Telegram.

В качестве реальных пользователей разрабатываемого решения можно выделить молодых и взрослых русскоговорящих пользователей Telegram.

Местом выполнения работ является жилое помещение со столом, роутером, компьютером и его периферией.

В современном мире исследователи определяют современное обществе как «информационное общество», «сетевое общество» и др. Все это указывает на то, что в обществе глобальную роль стала играть информация и способы, и средства её распространения. Личные и социальные коммуникации с целью развлечения и делового сотрудничества все чаще уходят в сеть и распространяются при помощи информационно-коммуникационных технологий. Поэтому можно заключить что разработка социальных ботов является актуальной за счет возможности использования их с целью социализации и коммуникации людей.

5.2. Правовые и организационные вопросы обеспечения безопасности

Правовые нормы трудового законодательства

5.2.1. Режим рабочего времени

В соответствии со статьей 100 ТК РФ, режим рабочего времени предусматривает продолжительность рабочей недели. При пятидневной рабочей неделе предусматривается два выходных дня. Длительность рабочей смены не более 8 часов. Также устанавливается два регламентированных перерыва (не менее 20 минут после 1-2 часов работы и не менее 30 минут после 2 часов работы). Также предусмотрен обеденные перерыв не менее 40 минут.

5.2.2. Защита персональных данных работника

В соответствии со статьей 86 ТК РФ, обработка персональных данных может осуществляться исключительно в целях обеспечения соблюдения законов и иных нормативных правовых актов. Все персональные данные о работнике следует получать от него же.

5.2.2.1. Эргономические требования к правильному расположению и компоновке рабочей зоны

Согласно ГОСТ 12.2.032-78, рабочее место для выполнения работ сидя организуют при легкой работе, не требующей свободного передвижения работающего. Конструкция рабочего места и взаимное расположение всех его элементов должны соответствовать антропометрическим, физиологическим и психологическим требованиям, а также характеру работы. Также рабочее место должно быть организовано в соответствии с требованиями стандартов, технических условий и (или) методических указаний по безопасности труда.

5.3.Производственная безопасность

При работе с электрическими приборами присутствую различные вредные факторы, которые негативно воздействуют на организм человека. Также при взаимодействии с электрическими приборами возникают опасные факторы, их перечень представлен в таблице №6

Таблица 6 - Производственная безопасность

Факторы (ГОСТ 12.0.003-2015)	Этапы работ			Нормативные документы
	Разработка	Изготовление	Эксплуатация	
Отклонение показателей микроклимата в помещении	-	-	+	СП 1.2.3685-21
Недостаток естественного освещения	-	-	+	СП 52.13330.2016

Недостаточная освещенность рабочей зоны	-	-	+	СП 52.13330.2016

5.3.1. Отклонение показателей микроклимата в помещении

Микроклимат является важной характеристикой производственных помещений. В организме человека происходит непрерывное выделение тепла. Одновременно с процессами выделения тепла происходит непрерывная теплоотдача в окружающую среду. Равновесие между выделением тепла и теплоотдачей регулируется процессами терморегуляции, т.е. способностью организма поддерживать постоянство теплообмена с сохранением постоянной температуры тела. Отдача тепла происходит различными видами: излучением, конвекцией, испарение влаги. Нарушение теплового баланса в условиях высокой температуры может привести к перегреву тела, и как следствие к тепловым ударам с потерей 44 сознания. В условиях низкой температуры воздуха возможно переохлаждение организма, могут возникнуть простудные болезни, радикулит, бронхит и другие заболевания. Показателями, которые характеризуют микроклимат рабочей зоны, являются:

- температура воздуха, ºС;
- относительная влажность воздуха, %;
- скорость движения воздуха, м/с.

Оптимальные значения этих характеристик зависят от сезона (холодный, тёплый), а также от категории физической тяжести работы. Для математика и программиста она является лёгкой (1а), так как работа проводится сидя, без систематических физических нагрузок. Оптимальные показатели микроклимата рабочей зоны, согласно СП1.2.3685-21, представлены в таблице №7, допустимые – в таблице №8.

Таблица 7 - Оптимальные показатели микроклимата

Период года	Температура воздуха, ºС	Относительная влажность воздуха, %	Скорость движения воздуха, м/с
Теплый	23-25	40-60	0,1
Холодный	22-24	40-60	0,1

Таблица 8 – Допустимые показатели микроклимата

Период года	Температура воздуха, ºС		Относительная влажность воздуха, %	Скорость движения воздуха, м/с	
	Диапазон ниже оптимальных величин	Диапазон выше оптимальных величин		Диапазон ниже оптимальных величин	Диапазон выше оптимальных величин
Теплый	21,0-22,9	24,1-25,0	15-75	0,1	0,2
Холодный	20,0-21,9	25,1-28,0	15-75	0,1	0,1

Мероприятия по доведению микроклиматических показателей до нормативных значений включаются в комплексные планы предприятий по охране труда. Это такие мероприятия, как:

– механизация и автоматизация производственных процессов, дистанционное управление ими;

– применение технологических процессов и оборудования, исключающих образование вредных веществ или попадания их в рабочую зону;

– установка систем вентиляции, кондиционирования, отопления.

К мероприятиям по оздоровлению воздушной среды в помещении относятся правильная организация вентиляции и кондиционирования воздуха, отопление помещений. Вентиляция может осуществляться естественным и

механическим путём. В зимнее время в помещении необходимо предусмотреть систему отопления.

По степени физической тяжести работа программиста относится к категории лёгких работ.

Таким образом, делаем вывод о том, что рабочее место программиста соответствует нормам показателей микроклимата так, как есть соответствие температурным показателям в помещении в холодный период – 23,3 °С, и в теплый период – 24,6 °С. Относительная влажность воздуха составляет 58%, что соответствует диапазону.

5.3.2. Недостаток естественного освещения

Помещения с постоянным пребыванием людей должны иметь естественное освещение. Без естественного освещения допускается проектировать помещения с временным пребыванием людей, помещения, которые определены соответствующими сводами правил и стандартами организаций на проектирование зданий и сооружений, а также помещения, размещение которых разрешено в подвальных этажах зданий и сооружений.

В учебных и учебно-производственных помещениях общеобразовательных организаций, интернатов, профессиональных образовательных организаций при одностороннем боковом освещении нормируемое значение коэффициента естественной освещенности должно быть обеспечено в расчетной точке, расположенной на пересечении вертикальной плоскости характерного разреза помещений и условной рабочей поверхности на расстоянии 1,2 м от стены, наиболее удаленной от световых проемов и иметь значение не более 0,5%. Равномерность естественного освещения не нормируется для производственных помещений с боковым освещением. В жилом помещении имеется одно окно с одной стороны, чего достаточно для этого помещения, так как в нем имеется искусственное освещение.

5.3.3. Недостаточная освещенность рабочей зоны

Производственное освещение является неотъемлемым элементом условий трудовой деятельности человека. При правильно организованном освещении

рабочего места обеспечивается сохранность зрения человека и нормальное состояние его нервной системы, а также безопасность в процессе производства. Производительность труда и качество выпускаемой продукции находятся в прямой зависимости от освещения. При неудовлетворительном освещении ощущается усталость глаз и переутомление, что приводит к снижению работоспособности. В ряде случаев это может привести к головным болям. Головные боли также могут быть вызваны пульсацией освещения, что в основном является результатом использования электромагнитных пускорегулирующих аппаратов (ПРА) для газоразрядных ламп, работающих на частоте 50 Гц. Для характеристики естественного освещения используется коэффициент естественной освещенности (КЕО). Величины КЕО для различных помещений лежат в пределах 0,1-12%.

Освещённость на рабочем месте должна соответствовать характеру зрительной работы; равномерное распределение яркости на рабочей поверхности и отсутствие резких теней; величина освещения постоянна во времени (отсутствие пульсации светового потока); оптимальная направленность светового потока и оптимальный спектральный состав; все элементы осветительных установок должны быть долговечны, взрыво-, пожаро-, электробезопасны.

Работа с приборами относится к зрительным работам средней точности для помещений жилых и общественных зданий. Согласно СП 52.13330.2016, такие помещения должны удовлетворять требованиям, указанным в таблице №9.

Таблица 9 - Требования для работы с приборами

Характеристика	Наименьший или	Разряд зрите	Подразряд зрите	Относительная продолж	Искусственное освещение		Естественное освещение
							КЕО $e_н$, %, при

зрительной работы	эквивалентный размер объекта различения, мм	льной работы	льный работы	ительность зрительной работы при направлении зрения на рабочую поверхность, %	Освещенность на рабочей поверхности от системы общего освещения, ЛК	Коэффициенты пульсации освещенности КП, %, не более	Верхнем или комбинированном	Боковом
							2,0	0,5
Высокой точности	Более 0,5	В	1	Не менее 70	150	20	2,0	0,5
			2	Менее 70	100	20	2,0	0,5

На рабочем месте соблюдаются необходимые нормы освещенности согласно ГОСТ Р 55710-2013. «Освещение рабочих мест внутри зданий. Нормы и методы измерений».

5.4.Экологическая безопасность

В настоящее время, когда встает проблема рационального использования природных ресурсов, охраны окружающей среды, уделяется большое внимание организации разумного воздействия на природу. Необходимо совершенствовать технологические процессы с целью сохранения окружающей среды от вредных выбросов.

В связи с тем, что основным средством работы являются средства измерения и электрические приборы, серьезной проблемой является электропотребление. Это влечет за собой общий рост объема потребляемой электроэнергии. Для удовлетворения потребности в электроэнергии, приходиться увеличивать мощность и количество электростанций. Соответственно, рост энергопотребления приводит к таким экологическим нарушениям, как глобальное потепление климата, загрязнение атмосферы и водного бассейна Земли вредными и ядовитыми веществами, опасность аварий в ядерных реакторах, изменение ландшафта Земли. Целесообразным является разработка и внедрение систем с малым потреблением энергии.

В жилом помещении не ведется никакого производства. К отходам, производимым в помещении можно отнести бытовой мусор.

Сточные воды здания относятся к бытовым сточным водам. За их очистку отвечает городской водоканал.

Основной вид мусора – это отходы печати, бытовой мусор (в т. ч. люминесцентные лампы), неисправное электрооборудование, коробки от техники, использованная бумага. Утилизация отходов печати вместе с бытовым мусором происходит в обычном порядке.

Утилизация средств измерений и электрических приборов согласно ФЗ от 24.06.1998 №89-ФЗ (ред. от 25.12.2018) «Об отходах производства и потребления» и распоряжению Правительства РФ от 25.07.2017 №1589-р осуществляется сотрудниками лаборатории и предусматривает следующие этапы:

1.Правильное заполнение акта списания с указанием факта невозможности дальнейшей эксплуатации перечисленной в акте измерительной техники, о чем имеется акт технического осмотра;

2.Осуществление списания перечисленной в акте измерительной техники с баланса предприятия с указанием в бухгалтерском отчете, так как утилизация возможна для осуществления только после окончательного списания;

3.Непосредственно утилизация измерительной техники с полным демонтажем устройств на составляющие детали с последующей сортировкой по видам материалов и их дальнейшей передачей на перерабатывающие заводы.

5.5.Безопасность в чрезвычайных ситуациях

Чрезвычайными ситуациями в подобных помещениях могут быть пожары. Основы пожарной безопасности определены по ГОСТ 12.1.004-91 и ГОСТ 12.1.010-76. Все производства по пожарной опасности подразделяются на 5 категорий: А, Б, В, Г, Д. Помещение, в котором будет выполняться работа, относится к категории В.

Причинами пожара могут быть:

– токи короткого замыкания;

– электрические перегрузки;

– выделение тепла, искрение в местах плохих контактов при соединении проводов;

– курение в неположенных местах.

Тушение горящего электрооборудования под напряжением должно осуществляются имеющимися огнетушителями ОУ-5. Чтобы предотвратить пожар в помещении, необходимо:

– содержать помещение в чистоте, убирать своевременно мусор. По окончании работы поводиться влажная уборка всех помещений;

– работа должна проводиться только при исправном электрооборудовании;

– на видном месте должен быть вывешен план эвакуации из помещения с указанием оборудования, которое нужно эвакуировать в первую очередь;

– уходящий из помещения последним должен проверить выключены ли нагревательные приборы, электроприборы и т.д. и отключение силовой и осветительной электрической сети.

Также необходимо соблюдение организационных мероприятий:

– правильная эксплуатация приборов, установок;

– правильное содержание помещения; – противопожарный инструктаж сотрудников аудитории;

– издание приказов по вопросам усиления ПБ;

– организация добровольных пожарных дружин, пожарнотехнических комиссий;

– наличие наглядных пособий и т.п.

В случаях, когда не удается ликвидировать пожар самостоятельно, необходимо эвакуироваться вслед за сотрудниками по плану эвакуации и ждать приезда специалистов, пожарников. При возникновении пожара должна сработать система пожаротушения, издав предупредительные сигналы, и передав на пункт пожарной станции сигнал о ЧС, в случае если система не сработала, по каким-либо причинам, необходимо самостоятельно произвести вызов пожарной службы по телефону 101, сообщить место возникновения ЧС и ожидать приезда специалистов.

5.6. Выводы по разделу «Социальная ответственность»

Таким образом можно сделать вывод по всему разделу «Социальная ответственность», что на рабочем месте показатели микроклимата, освещенности и электромагнитных излучений находятся в норме, что говорит о безопасности для человека. Анализ выявленных опасных факторов рабочего помещения показал, что электробезопасность, пожаробезопасность и экологическая безопасность находятся под контролем, и все необходимые меры для обеспечения безопасности принимаются.

Заключение

В данной работе был произведен анализ сферы использования интеллектуальных чат-ботов.

Разработан и программно реализован алгоритм для работы с социальным ботов, предложена рекомендательная система на основании тэгов, используемая в боте для формирования очереди профилей.

Выбран стек информационных технологий для реализации социального чат-бота - Python, API Telegram, PostgreSQL.

Для тестирования сценария использования предложенного социального бота был спроектирован пользовательский интерфейс, спроектирована, реализована и заполнена пользовательскими данными база данных PostgreSQL. В работе приведено описано поведение бота при различных взаимодействиях с пользователями – регистрация, поиск профилей, редактирование профилей, удаление профиля.

Проведено тестирование работоспособности бота (единовременное подключение 20 пользователей), в результате которого были выявлены и устранены программные ошибки.

В разделе стартапа был произведен анализ существующих решений и описано предложенное решение, указана бизнес-модель, целевая аудитория, приведены преимущества использования социального чат-бота.

Список публикаций студента

1. Brylina Irina V., Okonskaya Natalia K., Ermakov Mikhail A., Brylin Artem V. Education of the future in the conditions of VUCA-world // European Proceedings of Social and Behavioural Sciences. EpSBS. II International Conference on Economic and Social Trends for Sustainability of Modern Society: ICEST 2021. – p. 1372-1380. – Режим доступа: https://www.elibrary.ru/item.asp?id=47231987

2. Брылина И.В., Брылин А.В. Университет: устойчивость vs изменчивость // Сборник «Этика меняющегося мира: теория, практика, технологии»: Материалы VIII Всероссийской (национальной) научно-практической конференции. – Красноярск, 2020. – с. 6-8. – Режим доступа: https://elibrary.ru/download/elibrary_44734358_54615151.pdf

3. Брылина И.В., Брылин А.В. Пути интеграции предпринимательского университета с бизнесом и властью в кластерно-сетевое партнерство // Современное образование: интеграция образования, науки, бизнеса и власти. Материалы международной научно-методической конференции. В 2-х частях. Ч.2. 2022. - Томск: Издательство Томский государственный университет систем управления и радиоэлектроники. – с. 212-218. – Режим доступа: https://www.elibrary.ru/item.asp?id=48195483 ; https://www.elibrary.ru/download/elibrary_48195483_17046029.pdf

4. Солтангазин Е.Н., Брылин А.В., Иванов В.В. Создание геймифицированного продукта для адаптации иностранных/иногородних студентов в Томске // Научная инициатива иностранных студентов и аспирантов российских вузов, 2022. – В печати. – Режим доступа: http://iie.tpu.ru/sci_conf/results_2022/results_4.html

Список использованных источников

1. 10 главных трендов маркетинга в 2017 году // Ibusiness [Электронный ресурс]. – URL: http://ibusiness.ru/blog/future/42397 (дата обращения: 18.05.2022).

2. Алымов А.С., Баранюк В.В., Смирнова О.С. Детектирование бот-программ, имитирующих поведение людей в социальной сети «ВКонтакте» // International Journal of Open Information Technologies. – 2016. – Vol. 4, № 8. – p. 55-60.

3. Баева Н. Книга 5. Непрерывное улучшение. [Электронный ресурс]. – URL: https://is59-2015.susu.ru/natashab/2018/05/03/kniga-5-nepreryivnoe-uluchshenie/ (дата обращения: 18.05.2022).

4. Белоус Е. Как чат боты создают ценность для вашего бизнеса // Энциклопедия маркетинга [Электронный ресурс]. – URL: http://www.marketing.spb.ru/lib-comm/dm/bot.htm (дата обращения: 18.05.2022).

5. Боты в Системе взаимодействия. / 1С. Заметки из Зазеркалья. 18.05.2020 – URL: https://wonderland.v8.1c.ru/blog/boty-v-sisteme-vzaimodeystviya/ (дата обращения 18.05.2022)

6. Василькова В.В., Легостаева Н.И. Боты на публичных аренах социальных сетей // Социологический журнал. 2021. – Том 27. – № 4. – с. 99-117. – URL: https://doi.org/10.19181/socjour.2021.27.4.8647 (дата обращения 18.05.2022)

7. Василькова В.В., Легостаева Н.И. Социальные боты в политической коммуникации // Вестник российского университета дружбы народов. серия: Социология. – 2019. – Том 19. – № 1. – с. 121-133. – URL: https://doi.org/10.22363/2313-2272-2019-19-1-121-133 (дата обращения 18.05.2022)

8. Василькова В.В., Легостаева Н.И., Радушевский В.Б. Социальные боты как инструмент развития гражданского участия // Мониторинг общественного мнения: Экономические и социальные перемены. – 2019. – № 5. – с. 19-42. – URL: https://doi.org/10.14515/monitoring.2019.5.02 (дата обращения 18.05.2022)

9. ГОСТ 3.1105-2011 Единая система технологической документации (ЕСТД). Формы и правила оформления документов общего назначения

10. ГОСТ 19.101-77 Единая система программной документации. Виды программ и программных документов

11. ГОСТ 19.401-78 Единая система программной документации. Текст программы. Требования к содержанию и оформлению

12. ГОСТ 19.402-78 Единая система программной документации. Описание программы

13. ГОСТ 19.502-78 Единая система программной документации. Описание применения. Требования к содержанию и оформлению

14. ГОСТ 19.504-79 Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению

15. ГОСТ 19.701-90 Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения

16. Зильберман Н. Н. Технологии виртуальных собеседников и формы речевого взаимодействия // Гуманитарная информатика. – 2009. – № 5. – с. 80-85.

17. Иванова Е. Г. Интеллектуальные диалоговые интерфейсы в системах электронной коммерции // Известия ЮФУ. Технические науки. – 2007. – № 2. – С. 42-52.

18. Иванов А. Д. Чат-боты в Telegram и в контакте как новый канал распространения новостей // Вестник Волжского университета им. В.Н. Татищева. 2016. – Т. 1. – № 3. – с. 126-132.

19. Кавеева А.Д., Гурин К.Е. Искусственные профили «ВКонтакте» и их влияние на социальную сеть пользователей // Журнал социологии и социальной антропологии. – 2018. – №21(2). – с. 214-231. – URL: https://doi.org/10.31119/jssa.2018.21.2.8 (дата обращения: 18.05.2022).

20. Кузнецов В. В. Перспективы развития чат-ботов//Успехи современной науки. – 2016. – № 12. – с. 16-19.

21. Орлова Э.А., Шапинская Е.Н., Урмина И.А., Каменец А.В. Социальное взаимодействие как процесс. – Москва: ГАСК, 2013. – 304 с.

22. Смыслова Л.В. Чат-бот как современное средство интернет-коммуникаций / Л. В. Смыслова. – Текст: непосредственный // Молодой ученый. – 2018. – № 9 (195). – С. 36-39. – URL: https://moluch.ru/archive/195/48623/ (дата обращения: 18.05.2022).

23. Чат-боты в укреплении взаимодействия между сообществами-поколениями социальных сетей / Никитина С.С., Берестнева О.Г., Труфанов А.И. [и др.] // Информационные технологии в науке, управлении, социальной сфере и медицине: сборник научных трудов IV Международной научной конференции, 5-8 декабря 2017 г., Томск: в 2 ч. – Томск: Изд-во ТПУ, 2017. – Ч. 2. – с. 64-67. – URL: http://earchive.tpu.ru/handle/11683/46946 (дата обращения: 18.05.2022).

24. Чивилёв А. А. Межличностная коммуникация с виртуальными собеседниками в пространстве современной культуры // Культурология и искусствоведение: материалы II Междунар. науч. конф. – Казань: Бук, 2016. – с. 10-13.

25. Чесноков В.О. Применение алгоритма выделения сообществ в информационном противоборстве в социальных сетях // Вопросы кибербезопасности. – 2017. – №1 (19). – с. 37-44. – DOI: 10.21681/2311-3456-2017-1-37-44 (дата обращения: 18.05.2022).

26. Shawar B. A., Atwell E. A comparison between ALICE and Elizabeth chatbot systems. – University of Leeds, School of Computing research report 2002.19, 2002. – 23 p. – URL: https://eprints.whiterose.ac.uk/81930/1/AComparisonBetweenAliceElizabeth.pdf (дата обращения: 18.05.2022).

27. Shawar B. A., Atwell E. Chatbots: are they really useful? // LDV Forum. – 2007. – V. 22. – №. 1. – p. 29-49. – URL: http://ibusiness.ru/blog/future/42397 (дата обращения: 18.05.2022).

28. Аудитория российских мессенджеров – URL: https://www.statista.com/statistics/1065027/russia-messenger-audience-by-

app/#:~:text=Telegram%2C%20which%20was%20unblocked%20by,at%20least%20once%20a%20month. (дата обращения 18.05.2022)

29. Главные отличия самых популярных мессенджеров – URL: https://trends.rbc.ru/trends/industry/6156fef89a7947827bf5a9b7 (дата обращения 18.05.2022)

30. Рынок дейтинга в России– URL: https://vc.ru/services/167105-rynok-onlayn-znakomstv-proshloe-nastoyashchee-budushchee (дата обращения 18.05.2022)

Приложение А

Код бота. main.py – основная логика работы с ботом, userinterface.py – клавиатуры и функции для взаимодействия с пользовательским графическим интерфейсом, config.py -–настройки локального сервера базы данных.

Main.py

```
import telebot
import requests
import anketa

import psycopg2
from config import host, user, password, db_name

from userinterface import *
from bd import checkIfUserExist
global message

#tagsuser=[]

API_TOKEN = "сделайте свой)"
bot = telebot.TeleBot("сделайте свой)")
global USERINCHATID
#MYUSERID = bot.user.id

@bot.message_handler(commands=['help'])
def send_about(message):
    text = 'Бот рекомендует тебе профили людей со схожими интересами'
    bot.send_message(message.from_user.id, text=text)

@bot.message_handler(commands=['start'])
def send_welcome(message):
    global USERINCHATID
    USERINCHATID = message.from_user.id
    anketa.telegramID = USERINCHATID
    temp = (message.from_user.id) ##айдишник в телеге
    connection = psycopg2.connect(
        host=host,
        user=user,
        password=password,
        dbname=db_name
        # database = db_name
        )
```

```
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT telegramID FROM users WHERE telegramid = %s", (temp,)
)
                if cursor.fetchone() is None:
                    text = 'Привет, ты еще не создавал профиль, давай создадим'
                    bot.send_message(message.from_user.id, text)
                    addInitialKeyboard(message)
                else:
                    text = 'Привет, я тебя помню, продолжим?'
                    bot.send_message(message.from_user.id, text = text)

                    mainkeyboard = types.ReplyKeyboardMarkup(row_width=3,
resize_keyboard=True)
                    button_searchProfiles = types.KeyboardButton(text="Поиск других
профилей")
                    button_changeProfile = types.KeyboardButton(text="Изменить
профиль")
                    button_quitProfile = types.KeyboardButton(text="Выключить
профиль")
                    mainkeyboard.add(button_searchProfiles, button_changeProfile,
button_quitProfile)
                    bot.send_message(message.from_user.id, text= 'Выбери что хочешь
делать', reply_markup=mainkeyboard)
                    #bot.send_photo(message.from_user.id, downloaded_file,
caption=showInfoNotMedia(message),
                    #               reply_markup=mainkeyboard)
            connection.close()

        @bot.message_handler(regexp='Создать профиль')
        def createProfile(message):
            if anketa.callback == 'Создать профиль':
                global USERINCHATID
                msg = bot.send_message(message.from_user.id, 'Введи свое имя')
                bot.register_next_step_handler(msg, addName)
            else:
                anketa.callback = 'Ошибка1'
                bot.send_message(USERINCHATID, 'Извини, бот сломавса1 :(')
                bot.send_message(message.from_user.id, 'Извини, бот сломавса1 :(')

        def addName(message):
            anketa.name = message.text
            connection = psycopg2.connect(
                host=host,
```

```
                user=user,
                password=password,
                dbname=db_name
                # database = db_name
                )
            bot.user.username
            with connection.cursor() as cursor:
               cursor.execute(
               "INSERT INTO users (telegramid, name, telegramusername) VALUES
(%s, %s, %s);", ( (message.from_user.id,), message.text,
message.from_user.username))
            connection.commit()
            connection.close()
            msg = bot.send_message(message.from_user.id, 'Введи свой возраст')
            bot.register_next_step_handler(msg, addAge)

        def addAge(message):
           # bot.send_message(message.from_user.id, 'Введи свой возраст')
            anketa.age = message.text
            connection = psycopg2.connect(
               host=host,
               user=user,
               password=password,
               dbname=db_name
               # database = db_name
                  )
            with connection.cursor() as cursor:
               cursor.execute(
               "UPDATE users SET age = %s WHERE telegramid = %s;",
((int(message.text),), (message.from_user.id,)) )
            connection.commit()
            connection.close()
            #msg = addGenderKeyboard(message)
            keyboard = addGenderKeyboard(message)

            #msg = bot.send_message(message.chat.id, text='Выбери свой пол',
reply_markup=keyboard)
            #msg = bot.send_message(message.from_user.id, 'Выбери какого ты пола')
            #bot.register_next_step_handler(msg, addGender)
            bot.register_next_step_handler(message, addGender)

        def addGender(message):
           # bot.send_message(message.from_user.id, 'Введи свой возраст')
            connection = psycopg2.connect(
```

```
                host=host,
                user=user,
                password=password,
                dbname=db_name)
                # database = db_name)
            with connection.cursor() as cursor:
                if message.text == "Парень":
                    cursor.execute(
                    "UPDATE users SET gender = 'M' WHERE telegramid = %s;",
(message.from_user.id,) )
                elif message.text == "Девушка":
                    cursor.execute(
                    "UPDATE users SET gender = 'Ж' WHERE telegramid = %s;",
(message.from_user.id,) )
            connection.commit()
            connection.close()
            keyboard = addgenderToSearchKeyboard(message)


            #msg = bot.send_message(message.chat.id, text='Кого будем искать?',
reply_markup=keyboard)
            #message.text
            #msg.text
            #msg = bot.send_message(message.from_user.id, 'Кого будем искать?',
reply_markup=keyboard)
            #bot.register_next_step_handler(msg, genderToSearch)
            bot.register_next_step_handler(message, genderToSearch)

            # if message.text == "Парень":
            #     anketa.gender = "M"
            # elif message.text == "Девушка":
            #     anketa.gender = "Ж"

        # msg = bot.send_message(message.from_user.id, 'Кого будем искать?',
reply_markup= addgenderToSearchKeyboard(message))
        # bot.register_next_step_handler(msg,genderToSearch)
            #msg = bot.send_message(message.from_user.id, 'Введи свой город')
            #bot.register_next_step_handler(msg, addCity)

        def genderToSearch(message):
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
```

```
            dbname=db_name)
        # database = db_name)
        with connection.cursor() as cursor:
            if message.text == "Парней":
                cursor.execute(
                    "UPDATE users SET searchgender = 'M' WHERE telegramid =
%s;", (message.from_user.id,) )
            elif message.text == "Девушек":
                cursor.execute(
                    "UPDATE users SET searchgender = 'Ж' WHERE telegramid =
%s;", (message.from_user.id,) )
            elif message.text == "Всех":
                cursor.execute(
                    "UPDATE users SET searchgender = 'All' WHERE telegramid =
%s;", (message.from_user.id,) )
        connection.commit()
        connection.close()

        msg = bot.send_message(message.from_user.id, 'Введи свой город')
        bot.register_next_step_handler(msg, addCity)

    def addCity(message):
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            dbname=db_name)
        # database = db_name)
        with connection.cursor() as cursor:
            cursor.execute(
                "UPDATE users SET city = %s WHERE telegramid = %s;",
((message.text),(message.from_user.id,)) )
        connection.commit()
        connection.close()

        # bot.send_message(message.from_user.id, 'Введи свой возраст')
        ###bot.send_message(message.from_user.id, 'Ты из ' + str(anketa.city))
        ###tags
        tagskeyboard = types.InlineKeyboardMarkup(row_width=2)
        tagSportButton = types.InlineKeyboardButton(text="Спорт",
callback_data="Спорт")
        tagMusicButton = types.InlineKeyboardButton(text="Музыка",
callback_data="Музыка")
```

```
            tagMovieButton = types.InlineKeyboardButton(text="Фильмы",
callback_data="Фильмы")
            tagTalkButton = types.InlineKeyboardButton(text="Пообщаться",
callback_data="Пообщаться")
            tagHangoutButton = types.InlineKeyboardButton(text="Погулять",
callback_data="Погулять")
            tagEducationButton = types.InlineKeyboardButton(text="Образование",
callback_data="Образование")
            tagsQuitButton  = types.InlineKeyboardButton(text="Закончить с выбором
тэгов", callback_data="Закончить")
            tagskeyboard.add(tagSportButton, tagMusicButton, tagMovieButton,
tagTalkButton, tagHangoutButton, tagEducationButton,tagsQuitButton)

            #tagsuser = []
            msg = bot.send_message(message.from_user.id,'Выбери 3 тэга для своего
профиля, затем нажми на кнопку закончить', reply_markup=tagskeyboard)
            bot.register_callback_query_handler(call, addTags)
            #bot.register_callback_query_handler(addTags)
            #print(tagsuser)
            ### bot.register_next_step_handler(msg, addTags)
            msg = bot.send_message(message.from_user.id, 'Напиши что будет в
описании твоего профиля')
            bot.register_next_step_handler(msg, addDescription)

        def addDescription(message):
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "UPDATE users SET description = %s WHERE telegramid = %s;",
((message.text), (message.from_user.id,)) )
            connection.commit()
            connection.close()
            # bot.send_message(message.from_user.id, 'Введи свой возраст')
            anketa.descripition = message.text
            #bot.send_message(message.from_user.id, 'Описание: ' +
str(anketa.descripition))
            msg = bot.send_message(message.from_user.id, 'Добавь пожалуйста твое
фото')
            bot.register_next_step_handler(msg, addMedia)
```

```
        def showInfoNotMedia(message):
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT name, age, city, description FROM users WHERE
telegramid = %s;", ((message.from_user.id,)) )
                data = cursor.fetchone()

            connection.close()
            infoNotMedia =  data[0] + ', ' + str(data[1]) + ', ' + data[2] + '\n' + data[3]
            return infoNotMedia
            #bot.send_message(message.from_user.id, text = infoNotMedia , text=
downloaded.file)
            #bot.send_message()

        @bot.callback_query_handler(func=lambda call:True)
        def addTags(call):
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            if call.data == "Спорт":
                #tagsuser.append(call.data)
                with connection.cursor() as cursor:
                    cursor.execute(
                        "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(USERINCHATID,) )
                    tempdatatags = cursor.fetchone()
                    if tempdatatags[0] == None:
                        tag1 = "Спорт"
                        cursor.execute("UPDATE users SET tag1 = %s WHERE telegramid
= %s;", (tag1 , (USERINCHATID,)) )
                    elif tempdatatags[0] != None and tempdatatags[1] == None:
                        tag2 = "Спорт"
                        cursor.execute("UPDATE users SET tag2 = %s WHERE telegramid
= %s;" , (tag2 , (USERINCHATID,)) )
```

```
                elif tempdatatags[0] != None and tempdatatags[1] != None and
tempdatatags[2] == None:
                    tag3 = "Спорт"
                    cursor.execute("UPDATE users SET tag3 = %s WHERE telegramid
= %s;", (tag3, (USERINCHATID,)) )
                else: print("bug)")
            connection.commit()
        if call.data == "Музыка":
            #tagsuser.append(call.data)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(USERINCHATID,) )
                tempdatatags = cursor.fetchone()
                if tempdatatags[0] == None:
                    tag1 = "Музыка"
                    cursor.execute("UPDATE users SET tag1 = %s WHERE telegramid
= %s;", (tag1 , (USERINCHATID,)) )
                elif tempdatatags[0] != None and tempdatatags[1] == None:
                    tag2 = "Музыка"
                    cursor.execute("UPDATE users SET tag2 = %s WHERE telegramid
= %s;" , (tag2 , (USERINCHATID,)) )
                elif tempdatatags[0] != None and tempdatatags[1] != None and
tempdatatags[2] == None:
                    tag3 = "Музыка"
                    cursor.execute("UPDATE users SET tag3 = %s WHERE telegramid
= %s;", (tag3, (USERINCHATID,)) )
                else: print("bug)")
            connection.commit()
        if call.data == "Фильмы":
            #tagsuser.append(call.data)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(USERINCHATID,))
                tempdatatags = cursor.fetchone()
                if tempdatatags[0] == None:
                    tag1 = "Фильмы"
                    cursor.execute("UPDATE users SET tag1 = %s WHERE telegramid
= %s;", (tag1, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] == None:
                    tag2 = "Фильмы"
                    cursor.execute("UPDATE users SET tag2 = %s WHERE telegramid
= %s;", (tag2, (USERINCHATID,)))
```

```
                elif tempdatatags[0] != None and tempdatatags[1] != None and
tempdatatags[2] == None:
                    tag3 = "Фильмы"
                    cursor.execute("UPDATE users SET tag3 = %s WHERE telegramid
= %s;", (tag3, (USERINCHATID,)))
                else:
                    print("bug)")
            connection.commit()
        if call.data == "Пообщаться":
            #tagsuser.append(call.data)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(USERINCHATID,))
                tempdatatags = cursor.fetchone()
                if tempdatatags[0] == None:
                    tag1 = "Пообщаться"
                    cursor.execute("UPDATE users SET tag1 = %s WHERE telegramid
= %s;", (tag1, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] == None:
                    tag2 = "Пообщаться"
                    cursor.execute("UPDATE users SET tag2 = %s WHERE telegramid
= %s;", (tag2, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] != None and
tempdatatags[2] == None:
                    tag3 = "Пообщаться"
                    cursor.execute("UPDATE users SET tag3 = %s WHERE telegramid
= %s;", (tag3, (USERINCHATID,)))
                else:
                    print("bug)")
            connection.commit()
        if call.data == "Погулять":
            #tagsuser.append(call.data)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(USERINCHATID,))
                tempdatatags = cursor.fetchone()
                if tempdatatags[0] == None:
                    tag1 = "Погулять"
                    cursor.execute("UPDATE users SET tag1 = %s WHERE telegramid
= %s;", (tag1, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] == None:
                    tag2 = "Погулять"
```

```
                    cursor.execute("UPDATE users SET tag2 = %s WHERE telegramid
= %s;", (tag2, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] != None and
tempdatatags[2] == None:
                    tag3 = "Погулять"
                    cursor.execute("UPDATE users SET tag3 = %s WHERE telegramid
= %s;", (tag3, (USERINCHATID,)))
                else:
                    print("bug)")
            connection.commit()
        if call.data == "Образование":
            tagsuser.append(call.data)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(USERINCHATID,))
                tempdatatags = cursor.fetchone()
                if tempdatatags[0] == None:
                    tag1 = "Образование"
                    cursor.execute("UPDATE users SET tag1 = %s WHERE telegramid
= %s;", (tag1, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] == None:
                    tag2 = "Образование"
                    cursor.execute("UPDATE users SET tag2 = %s WHERE telegramid
= %s;", (tag2, (USERINCHATID,)))
                elif tempdatatags[0] != None and tempdatatags[1] != None and
tempdatatags[2] == None:
                    tag3 = "Образование"
                    cursor.execute("UPDATE users SET tag3 = %s WHERE telegramid
= %s;", (tag3, (USERINCHATID,)))
                else:
                    print("bug)")

        connection.close()
        if call.data == "Закончить":
            msg = bot.send_message(USERINCHATID, 'Напиши что будет в
описании твоего профиля')
            bot.register_next_step_handler(msg, addDescription)


    @bot.message_handler(content_types='photo')
    def addMedia(message):
        #bot.send_message(message.from_user.id, text = 'Спасибо за
фотографию!')
```

```
            raw = message.photo[-1].file_id
            path = raw + ".jpg"
            file_info = bot.get_file(raw)
            downloaded_file = bot.download_file(file_info.file_path)

            with open(path, 'wb') as new_file:
                new_file.write(downloaded_file)

            global fotka
            fotka = downloaded_file

            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "UPDATE users SET photo = %s WHERE telegramid = %s;", (raw
,(message.from_user.id,)))
                connection.commit()
                # cursor.execute(
                #     "SELECT photo from users WHERE telegramid = %s;",
(message.from_user.id,) )
                # photoid = cursor.fetchone()
                # photo_info = bot.get_file(photoid)
                # #file_info = bot.get_file(raw)
                # downloaded_file = bot.download_file(photo_info.file_path)

            connection.close()
            #вставка названия фотки в бд
            #select названия фотки -> загрузка из папки
            bot.send_message(message.from_user.id, text='Так выглядит твой
профиль:')

            mainkeyboard = types.ReplyKeyboardMarkup(row_width=3,
resize_keyboard=True)
            button_searchProfiles = types.KeyboardButton(text="Поиск других
профилей")
            button_changeProfile = types.KeyboardButton(text="Изменить профиль")
            button_quitProfile = types.KeyboardButton(text="Выключить профиль")
```

```
        mainkeyboard.add(button_searchProfiles, button_changeProfile,
button_quitProfile)

        tagskeyboard = types.InlineKeyboardMarkup(row_width=6)

        ##tags
        connection = psycopg2.connect(
           host=host,
           user=user,
           password=password,
           dbname=db_name)
        # database = db_name)
        with connection.cursor() as cursor:
           cursor.execute(
              "SELECT tag1, tag2, tag3 FROM users WHERE telegramid = %s;",
(message.from_user.id,) )
           data = cursor.fetchone()
        if data[0] == None:
           amountoftags = 0 ##нет 1 тэга
        elif data[1] == None:
           amountoftags = 1 #нет 2 тэга
        elif data[2] == None:
           amountoftags = 2 #нет 3 тэга
        else: amountoftags = 3
        connection.close()
        ##tags

        for i in range (amountoftags):
           if data[i] == None:
              print("bug")
           else:
              tagButton = types.InlineKeyboardButton(text=data[i],
callback_data="calling bruv" )
              tagskeyboard.add(tagButton)

        #bot.send_message(message.from_user.id, text = "нда",
reply_markup=tagskeyboard)
        bot.send_photo(message.from_user.id, downloaded_file,
caption=showInfoNotMedia(message),reply_markup=tagskeyboard)
        bot.send_message(message.from_user.id, text = "Что хочешь делать
дальше?", reply_markup=mainkeyboard)
        #bot.send_photo(message.from_user.id, downloaded_file, caption =
showInfoNotMedia(message), reply_markup=(mainkeyboard,tagskeyboard ))
```

```
        @bot.message_handler(content_types='text', regexp="Поиск других
профилей")
        def searchProfiles(message):
            bot.send_message(message.from_user.id, text = 'Ищем других
пользователей!')

            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT name, age, gender, city, description, searchgender, tag1,
tag2, tag3, activestatus, telegramusername, telegramid, photo FROM users WHERE
telegramid = %s;", (message.from_user.id,) )
                myprofiledata = cursor.fetchone()##my profile

                myprofiledata[0]##name
                myprofiledata[1]##age
                myprofiledata[2]##gender
                myprofiledata[3]##city
                myprofiledata[4]##description
                myprofiledata[5]##searchgender
                myprofiledata[6]##tag1
                myprofiledata[7]##tag2
                myprofiledata[8]##tag3
                myprofiledata[9]##activestatus
                myprofiledata[10]##telegramusername
                myprofiledata[11]  ##telegramid
                myprofiledata[12] ##photo
                mytagset = set([myprofiledata[6],myprofiledata[7],myprofiledata[8]])
                mytags = list(mytagset)
                for tag in mytags:
                    if tag == None or tag == "None":
                        mytags.remove(tag)
                mytags = set(mytags)
                #mytagset.remove(None)

                ##убрать свой профиль из поиска
                cursor.execute(
                    "SELECT telegramid, name, age, gender, city, description,
searchgender, tag1, tag2, tag3, activestatus, telegramusername, photo FROM users
```

```
EXCEPT(SELECT telegramid, name, age, gender, city, description, searchgender,
tag1, tag2, tag3, activestatus, telegramusername, photo FROM users WHERE
telegramid = %s)", (message.from_user.id,)
                )
                query3tags=[]
                query2tags=[]
                query1tag=[]
                queryall=[]
                uselessquerry=[]
                sharedtags = []
                allQuerry = cursor.fetchall()
                for person in allQuerry:
                    person[0]  # telegramid
                    person[1]  # name
                    person[2]  # age imp
                    person[3]  # gender imp
                    person[4]  # city imp
                    person[5]  # description
                    person[6]  # searchgender imp
                    person[7]  # tag1 imp
                    person[8]  # tag2 imp
                    person[9]  # tag3 imp
                    person[10]  # activestatus
                    person[11] #telegramusername
                    person[12] #photo
                    personTagSet=set([person[7],person[8],person[9]])
                    persontags = list(personTagSet)
                    for tag in persontags:
                        if tag == None or tag == "None":
                            persontags.remove(tag)
                    personTagSet = set(persontags)
                    sharedtagsSet = mytagset.intersection(personTagSet)
                    sharedtags = list(sharedtagsSet)

                    temp = len(sharedtags) ## amount of mutual shared tags with me and
other person

                    if len(sharedtags) == 3:##tags compatability
                        if myprofiledata[5] == person[3] and (person[6] == myprofiledata[2]
or person[6] == "All"):##mygendertosearch//person gender ##gender compatability
between mysearchgender and person.gender and person.searchgender and my gender
                            if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:##age compatability
                                if myprofiledata[3] == person[4]: ##city compatability
```

```
                        query3tags.append(person)
                elif myprofiledata[5] == "All" and (person[6] == myprofiledata[2] or
person[6] == "All"):
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            query3tags.append(person)
            elif len(sharedtags) == 2:##2 mutual tags
                if myprofiledata[5] == person[3] and (person[6] == myprofiledata[2]
or person[6] == "All"):  ##mygendertosearch//person gender ##gender compatability
between mysearchgender and person.gender and person.searchgender and my gender
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            query2tags.append(person)
                elif myprofiledata[5] == "All" and (person[6] == myprofiledata[2] or
person[6] == "All"):
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            query2tags.append(person)
            elif len(sharedtags) == 1 :##1 mutual tag
                if myprofiledata[5] == person[3] and (person[6] == myprofiledata[2]
or person[6] == "All"):  ##mygendertosearch//person gender ##gender compatability
between mysearchgender and person.gender and person.searchgender and my gender
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            query1tag.append(person)
                elif myprofiledata[5] == "All" and (person[6] == myprofiledata[2] or
person[6] == "All"):
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            query1tag.append(person)
            elif len(sharedtags) == 0:##0 mutual tags
                if myprofiledata[5] == person[3] and (person[6] == myprofiledata[2]
or person[6] == "All"):  ##mygendertosearch//person gender ##gender compatability
between mysearchgender and person.gender and person.searchgender and my gender
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            queryall.append(person)
```

```
                elif myprofiledata[5] == "All" and (person[6] == myprofiledata[2] or
person[6] == "All"):
                    if myprofiledata[1] <= person[2] + 4 and myprofiledata[1] >=
person[2] - 4:  ##age compatability
                        if myprofiledata[3] == person[4]:  ##city compatability
                            queryall.append(person)
                else:  #закидывает вообщех всех кто не подошёл
                    uselessquerry.append(person)
            query3tags##3 mutual
            query2tags##2 mutual
            query1tag## 1 mutual
            queryall## 0 mutual but same info
            uselessquerry## incorrect users
            sharedtags
            ##бот присылает сообщение с данными первого 3 тэгового юзера, и
даёт кейбоард на взаимодействие с ним
            ##как только выбираешь какое-то действие на кейбоарде, то тогда
этот пользователь удаляется из очереди и закидывается следующий с
кейбоардом
            len(query3tags)
            len(query2tags)
            len(query1tag)
            len(queryall)
            len(uselessquerry)
            QUERRY = []
            if len(query3tags) != 0:
                QUERRY.extend(query3tags)
                #QUERRY.append(query3tags)
            if len(query2tags) != 0:
                QUERRY.extend(query2tags)
            if len(query1tag) != 0:
                QUERRY.extend(query1tag)
            if len(queryall) != 0:
                QUERRY.extend(queryall)

            if len(QUERRY) == 0:
                bot.send_message(myprofiledata[11], text = "Похоже тут нет никого
твоего возраста из твоего города, кто мог бы тебя заинтересовать :c(TODO)")

            if len(QUERRY) != 0:
                currentPerson = QUERRY[0]
                currentPersonTagsAmount = 0
                #генерю инлайн(кейбоард) тэги для полученного пользователя
                tagPersonKeyboard = types.InlineKeyboardMarkup(row_width=3)
```

```
                if currentPerson[7] != None:
                    currentPersonTagsAmount += 1
                    tag1PersonButton =
types.InlineKeyboardButton(text=currentPerson[7], callback_data='person tag1')
                    tagPersonKeyboard.add(tag1PersonButton)

                if currentPerson[8] != None:
                    currentPersonTagsAmount += 1
                    tag2PersonButton =
types.InlineKeyboardButton(text=currentPerson[8], callback_data='person tag2')
                    tagPersonKeyboard.add(tag2PersonButton)

                if currentPerson[9] != None:
                    currentPersonTagsAmount += 1
                    tag3PersonButton =
types.InlineKeyboardButton(text=currentPerson[9], callback_data='person tag3')
                    tagPersonKeyboard.add(tag3PersonButton)

                mytagset = set([myprofiledata[6], myprofiledata[7], myprofiledata[8]])
                mytags = list(mytagset)
                for tag in mytags:
                    if tag == None or tag == "None":
                        mytags.remove(tag)
                mytags = set(mytags)
                personTagSet = set([currentPerson[7], currentPerson[8],
currentPerson[9]])
                persontags = list(personTagSet)
                for tag in persontags:
                    if tag == None or tag == "None":
                        persontags.remove(tag)
                personTagSet = set(persontags)
                sharedtagsSet = mytagset.intersection(personTagSet)
                sharedtags = list(sharedtagsSet)
                #photo

                #cursor.execute(
                #   "SELECT photo from users WHERE telegramid = %s;",
(query3tags[0][12]) )
                photoid = currentPerson[12]
                #photoid = query3tags[0][12]
                photo_info = bot.get_file(photoid)
                downloaded_file = bot.download_file(photo_info.file_path)
                #photo
```

```
                bot.send_photo(myprofiledata[11], downloaded_file, caption = "У этого пользователя с тобой " + str(len(sharedtags)) + " общих тэга!\n"
                + currentPerson[1] + "," + str(currentPerson[2]) + " ," + currentPerson[4] + "\n" + currentPerson[5], reply_markup=tagPersonKeyboard )

                swipeKeyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
                sendLikeButton = types.KeyboardButton(text="Нравится")
                sendDislikeButton = types.KeyboardButton(text="Не нравится")
                swipeKeyboard.add(sendLikeButton,sendDislikeButton)

            connection.close()
            bot.send_message(myprofiledata[11], text="Как тебе этот профиль?", reply_markup=swipeKeyboard)
            bot.register_next_step_handler(message, profileInteraction, QUERRY)
          #connection.close()


    # person[0]  # telegramid
    # person[1]  # name
    # person[2]  # age imp
    # person[3]  # gender imp
    # person[4]  # city imp
    # person[5]  # description
    # person[6]  # searchgender imp
    # person[7]  # tag1 imp
    # person[8]  # tag2 imp
    # person[9]  # tag3 imp
    # person[10]  # activestatus
    # person[11]  # telegramusername
    # person[12]  # photo
    #def profileInteraction(message, query3tags, query2tags, query1tag, queryall, uselessquerry):
    @bot.message_handler(content_types='text', regexp="Нравится")
    def profileInteraction(message, QUERRY):
        if message.text == "Нравится":
            #получить данные мои(передать из прошлог)
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
```

```
                "SELECT name, age, gender, city, description, searchgender, tag1,
tag2, tag3, activestatus, telegramusername, telegramid, photo FROM users WHERE
telegramid = %s;",(message.from_user.id,))
                myprofiledata = cursor.fetchone()  ##my profile
            connection.close()

            continueSearchKeyboard =
types.ReplyKeyboardMarkup(resize_keyboard=True)
            nextProfileButton = types.KeyboardButton(text = "Следующий
профиль")
            quitSearchProfileButton = types.KeyboardButton(text = "Остановить
поиск")
            continueSearchKeyboard.add(nextProfileButton,quitSearchProfileButton)

            bot.send_message(myprofiledata[11], text = "Ищем дальше?",
reply_markup=continueSearchKeyboard)
            bot.register_next_step_handler(message, nextProfileView, QUERRY)
            #logic with requiered person
            lastProfile = QUERRY[0]
            QUERRY.pop(0)#####################################ZDES
            ##lastlikeid
            #получить данные мои(передать из прошлог)
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "UPDATE users SET lastmatchid = %s WHERE telegramid = %s;",
((myprofiledata[11],), (lastProfile[0],)) )
                connection.commit()
            connection.close()
            #lastlikeid

            ##TEMPO
            myPersonTagsAmount = 0
            # генерю инлайн(кейбоард) тэги для полученного пользователя
            tagPersonKeyboard = types.InlineKeyboardMarkup(row_width=3)
            if myprofiledata[6] != None:
                myPersonTagsAmount += 1
                tag1PersonButton =
types.InlineKeyboardButton(text=myprofiledata[6], callback_data='person tag1')
```

```
                tagPersonKeyboard.add(tag1PersonButton)

            if myprofiledata[7] != None:
                myPersonTagsAmount += 1
                tag2PersonButton =
types.InlineKeyboardButton(text=myprofiledata[7], callback_data='person tag2')
                tagPersonKeyboard.add(tag2PersonButton)

            if myprofiledata[8] != None:
                myPersonTagsAmount += 1
                tag3PersonButton =
types.InlineKeyboardButton(text=myprofiledata[8], callback_data='person tag3')
                tagPersonKeyboard.add(tag3PersonButton)
            ##TEMPO

            ##SHAREDTAGS
            mytagset = set([myprofiledata[6],myprofiledata[7],myprofiledata[8]])
            mytags = list(mytagset)
            for tag in mytags:
                if tag == None or tag == "None":
                    mytags.remove(tag)
            mytags = set(mytags)
            personTagSet = set([lastProfile[7], lastProfile[8], lastProfile[9]])
            persontags = list(personTagSet)
            for tag in persontags:
                if tag == None or tag == "None":
                    persontags.remove(tag)
            personTagSet = set(persontags)
            sharedtagsSet = mytagset.intersection(personTagSet)
            sharedtags = list(sharedtagsSet)
            ##SHAREDTAGS


            #
            # tagPersonKeyboard = types.InlineKeyboardMarkup(row_width=3)
            # tag1PersonButton = types.InlineKeyboardButton(text=myprofiledata[6],
callback_data='person tag1')
            # tagPersonKeyboard.add(tag1PersonButton)
            #
            # tag2PersonButton = types.InlineKeyboardButton(text=myprofiledata[7],
callback_data='person tag2')
            # tagPersonKeyboard.add(tag2PersonButton)
            #
```

```
            # tag3PersonButton = types.InlineKeyboardButton(text=myprofiledata[8],
callback_data='person tag3')
            # tagPersonKeyboard.add(tag3PersonButton)

            photoid = myprofiledata[12]
            photo_info = bot.get_file(photoid)
            downloaded_file = bot.download_file(photo_info.file_path)
            # photo
            bot.send_photo(lastProfile[0], downloaded_file, caption = "Твой
профиль понравился пользователю: \n" + myprofiledata[0] + ", "
            + str(myprofiledata[1]) + ", " + myprofiledata[3] +'\n' + myprofiledata[4]
+ "\n У вас " + str(len(sharedtags)) + " общих тэга!",
reply_markup=tagPersonKeyboard )

            swipeKeyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
            sendLikeButton = types.KeyboardButton(text="Лайк")
            sendDislikeButton = types.KeyboardButton(text="Дизлайк")
            swipeKeyboard.add(sendLikeButton, sendDislikeButton)

            bot.send_message(lastProfile[0], text = "Нравится профиль?",
reply_markup = swipeKeyboard)


        elif message.text == "Не нравится":
            tempo = QUERRY
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT name, age, gender, city, description, searchgender, tag1,
tag2, tag3, activestatus, telegramusername, telegramid, photo FROM users WHERE
telegramid = %s;",(message.from_user.id,))
                myprofiledata = cursor.fetchone()  ##my profile
            connection.close()

            continueSearchKeyboard =
types.ReplyKeyboardMarkup(resize_keyboard=True)
            nextProfileButton = types.KeyboardButton(text = "Следующий
профиль")
```

```
            quitSearchProfileButton = types.KeyboardButton(text = "Остановить
поиск")
            continueSearchKeyboard.add(nextProfileButton,quitSearchProfileButton)

            bot.send_message(myprofiledata[11], text = "Ищем дальше?",
reply_markup=continueSearchKeyboard)
            QUERRY.pop(0)
            bot.register_next_step_handler(message, stopSearch)
            bot.register_next_step_handler(message, nextProfileView, QUERRY)

            # connection = psycopg2.connect(
            #     host=host,
            #     user=user,
            #     password=password,
            #     dbname=db_name)
            # # database = db_name)
            # with connection.cursor() as cursor:
            #     cursor.execute(
            #         "SELECT name, age, gender, city, description, searchgender, tag1,
tag2, tag3, activestatus FROM users WHERE telegramid = %s;",
            #         (message.from_user.id,))
            #     myprofiledata = cursor.fetchone()  ##my profile
            # connection.close()
            # lastProfile = query3tags.remove()
            # pass

        @bot.message_handler(content_types='text', regexp="Следующий профиль"
)
        def nextProfileView(message, QUERRY):
            #lastProfile = QUERRY[0]
            ##lastlikeid
            # получить данные мои(передать из прошлог)
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT name, age, gender, city, description, searchgender, tag1,
tag2, tag3, activestatus, telegramusername, telegramid, photo FROM users WHERE
telegramid = %s;",
                    (message.from_user.id,))
```

```
            myprofiledata = cursor.fetchone()  ##my profile

        mainkeyboard = types.ReplyKeyboardMarkup(row_width=3,
resize_keyboard=True)
        button_searchProfiles = types.KeyboardButton(text="Поиск других
профилей")
        button_changeProfile = types.KeyboardButton(text="Изменить профиль")
        button_quitProfile = types.KeyboardButton(text="Выключить профиль")
        mainkeyboard.add(button_searchProfiles, button_changeProfile,
button_quitProfile)


        if not QUERRY :
            bot.send_message(myprofiledata[11], text= "Похоже все подходящие
тебе профили закончились, перейдем в главное меню!",
reply_markup=mainkeyboard)
        else:
            lastProfile = QUERRY[0]
            with connection.cursor() as cursor:
                cursor.execute(
                    "UPDATE users SET lastmatchid = %s WHERE telegramid = %s;",
((myprofiledata[11],), (lastProfile[0],)))
                connection.commit()
            connection.close()
            # lastlikeid

            ##TEMPO
            myPersonTagsAmount = 0
            # генерю инлайн(кейбоард) тэги для полученного пользователя
            tagPersonKeyboard = types.InlineKeyboardMarkup(row_width=3)
            if myprofiledata[6] != None:
                myPersonTagsAmount += 1
                tag1PersonButton =
types.InlineKeyboardButton(text=myprofiledata[6], callback_data='person tag1')
                tagPersonKeyboard.add(tag1PersonButton)

            if myprofiledata[7] != None:
                myPersonTagsAmount += 1
                tag2PersonButton =
types.InlineKeyboardButton(text=myprofiledata[7], callback_data='person tag2')
                tagPersonKeyboard.add(tag2PersonButton)

            if myprofiledata[8] != None:
                myPersonTagsAmount += 1
```

```
                tag3PersonButton =
types.InlineKeyboardButton(text=myprofiledata[8], callback_data='person tag3')
                tagPersonKeyboard.add(tag3PersonButton)
            ##TEMPO

            ##SHAREDTAGS
            mytagset = set([myprofiledata[6], myprofiledata[7], myprofiledata[8]])
            mytags = list(mytagset)
            for tag in mytags:
                if tag == None or tag == "None":
                    mytags.remove(tag)
            mytags = set(mytags)
            personTagSet = set([lastProfile[7], lastProfile[8], lastProfile[9]])
            persontags = list(personTagSet)
            for tag in persontags:
                if tag == None or tag == "None":
                    persontags.remove(tag)
            personTagSet = set(persontags)
            sharedtagsSet = mytagset.intersection(personTagSet)
            sharedtags = list(sharedtagsSet)
            ##SHAREDTAGS

            #
            # tagPersonKeyboard = types.InlineKeyboardMarkup(row_width=3)
            # tag1PersonButton = types.InlineKeyboardButton(text=myprofiledata[6],
callback_data='person tag1')
            # tagPersonKeyboard.add(tag1PersonButton)
            #
            # tag2PersonButton = types.InlineKeyboardButton(text=myprofiledata[7],
callback_data='person tag2')
            # tagPersonKeyboard.add(tag2PersonButton)
            #
            # tag3PersonButton = types.InlineKeyboardButton(text=myprofiledata[8],
callback_data='person tag3')
            # tagPersonKeyboard.add(tag3PersonButton)

            photoid = lastProfile[12]
            photo_info = bot.get_file(photoid)
            downloaded_file = bot.download_file(photo_info.file_path)
            # photo
            bot.send_photo(myprofiledata[11], downloaded_file,caption="У этого
пользователя с тобой " + str(len(sharedtags)) + " общих тэга!\n"
            + lastProfile[1] + "," + str(lastProfile[2]) + " ," + lastProfile[4] + "\n" +
lastProfile[5], reply_markup=tagPersonKeyboard)
```

```
            #  bot.send_photo(lastProfile[0], downloaded_file,caption="Твой профиль понравился пользователю: \n" + myprofiledata[0] + ", "+ str(myprofiledata[1]) + ", "
             # + myprofiledata[3] + '\n' + myprofiledata[4] + "\n У вас " + str(len(sharedtags)) + " общих тэга!", reply_markup=tagPersonKeyboard)

            swipeKeyboard = types.ReplyKeyboardMarkup(resize_keyboard=True)
            sendLikeButton = types.KeyboardButton(text="Нравится")
            sendDislikeButton = types.KeyboardButton(text="Не нравится")
            swipeKeyboard.add(sendLikeButton, sendDislikeButton)

            bot.send_message(myprofiledata[11], text="Нравится профиль?", reply_markup=swipeKeyboard)
            bot.register_next_step_handler(message, profileInteraction, QUERRY)

        @bot.message_handler(content_types='text', regexp="Лайк" )
        def answerRequest(message):
            if message.text == "Лайк":
                connection = psycopg2.connect(
                    host=host,
                    user=user,
                    password=password,
                    dbname=db_name)
                # database = db_name)
                with connection.cursor() as cursor:
                    cursor.execute(
                        "SELECT telegramid, lastmatchid, telegramusername FROM users Where telegramid = %s", (message.from_user.id,) )
                    data=cursor.fetchone()

                with connection.cursor() as cursor:
                    cursor.execute(
                        "SELECT telegramid, lastmatchid, telegramusername FROM users Where telegramid = %s", (data[1],) )
                    mydata=cursor.fetchone()

                connection.close()

                mainkeyboard = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
                button_searchProfiles = types.KeyboardButton(text="Поиск других профилей")
```

```
            button_changeProfile = types.KeyboardButton(text="Изменить
профиль")
            button_quitProfile = types.KeyboardButton(text="Выключить
профиль")
            mainkeyboard.add(button_searchProfiles, button_changeProfile,
button_quitProfile)

            bot.send_message(data[0], text= "Приятно вам пообщаться\n" + "t.me/"
+ mydata[2])
            bot.send_message(mydata[0], text= "Приятно вам пообщаться\n" +
"t.me/" + data[2])
            #bot.send_message(lastProfile[0], text = "Приятно вам пообщаться\n" +
"t.me/" + myprofiledata[10])
            #bot.send_message(lastProfile[0], text= "Держи его id" +
str(myprofiledata[0]) )
            #bot.send_message(myprofiledata[11], text= "Приятно вам
пообщаться\n" + "t.me/"+ lastProfile[11])
        elif message.text == "Дизлайк":

            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT telegramid, lastmatchid, telegramusername FROM users
Where telegramid = %s",(message.from_user.id,))
                data = cursor.fetchone()
            connection.close()

            mainkeyboard = types.ReplyKeyboardMarkup(row_width=3,
resize_keyboard=True)
            button_searchProfiles = types.KeyboardButton(text="Поиск других
профилей")
            button_changeProfile = types.KeyboardButton(text="Изменить
профиль")
            button_quitProfile = types.KeyboardButton(text="Выключить
профиль")
            mainkeyboard.add(button_searchProfiles, button_changeProfile,
button_quitProfile)
            bot.send_message(data[1], text="Переходим в главное меню",
reply_markup=mainkeyboard)
```

```
        def answerfunc(message, lastProfile, myprofiledata, negr):
            if message.text == "Лайк":
                bot.send_message(lastProfile[0], text = "Приятно вам пообщаться\n" +
"t.me/" + myprofiledata[10])
                #bot.send_message(lastProfile[0], text= "Держи его id" +
str(myprofiledata[0]) )
                bot.send_message(myprofiledata[11], text= "Приятно вам
пообщаться\n" + "t.me/"+ lastProfile[11])
            elif message.text == "Дизлайк":
                pass

            # myprofiledata[0]##name
            #       myprofiledata[1]##age
            #       myprofiledata[2]##gender
            #       myprofiledata[3]##city
            #       myprofiledata[4]##description
            #       myprofiledata[5]##searchgender
            #       myprofiledata[6]##tag1
            #       myprofiledata[7]##tag2
            #       myprofiledata[8]##tag3
            #       myprofiledata[9]##activestatus
            ##

        @bot.message_handler(content_types='text', regexp="Остановить поиск" )
        def stopSearch(message):
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "SELECT telegramid FROM users Where telegramid =
%s",(message.from_user.id,))
                data = cursor.fetchone()
            connection.close()
            mainkeyboard = types.ReplyKeyboardMarkup(row_width=3,
resize_keyboard=True)
            button_searchProfiles = types.KeyboardButton(text="Поиск других
профилей")
            button_changeProfile = types.KeyboardButton(text="Изменить профиль")
            button_quitProfile = types.KeyboardButton(text="Выключить профиль")
```

```
        mainkeyboard.add(button_searchProfiles, button_changeProfile,
button_quitProfile)
        bot.send_message(data[0], text = "Переходим в главное меню",
reply_markup=mainkeyboard)


    @bot.message_handler(content_types='text', regexp="Изменить профиль")
    def changeMyProfile(message):
        bot.send_message(message.from_user.id, text = 'Сейчас твой профиль
выглядит так:')
        bot.send_photo(message.from_user.id, fotka,
caption=showInfoNotMedia(message))
        msg = bot.send_message(message.from_user.id, text = 'Давай заполним
твой профиль заново, введи своё имя!')
        bot.register_next_step_handler(msg, changeName)

    @bot.message_handler(content_types='text', regexp="Выключить профиль")
    def searchProfiles(message):
        bot.send_message(message.from_user.id, text = 'Пока! Рад был с тобой
пообщаться, всегда можешь вернуться (IN DEVELOPMENT)')




    ###############UPDATE BLOCK
    @bot.message_handler(regexp='Изменить профиль')
    def changeName(message):
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            dbname=db_name)
        # database = db_name)
        with connection.cursor() as cursor:
            cursor.execute(
                "UPDATE users SET name = %s WHERE telegramid = %s;",
((message.text), (message.from_user.id,)))
        connection.commit()
        connection.close()

        msg = bot.send_message(message.from_user.id, 'Введи свой возраст')
        bot.register_next_step_handler(msg, changeAge)

    def changeAge(message):
```

```
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "UPDATE users SET age = %s WHERE telegramid = %s;",
((message.text), (message.from_user.id,)))
            connection.commit()
            connection.close()
            #anketa.age = message.text
            msg = addGenderKeyboard(message)
            bot.register_next_step_handler(msg, changeGender)

        def changeGender(message):
            connection = psycopg2.connect(
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                if message.text == "Парень":
                    cursor.execute(
                        "UPDATE users SET gender = 'M' WHERE telegramid = %s;",
(message.from_user.id,))
                elif message.text == "Девушка":
                    cursor.execute(
                        "UPDATE users SET gender = 'Ж' WHERE telegramid = %s;",
(message.from_user.id,))
            connection.commit()
            connection.close()

            msg = bot.send_message(message.from_user.id, 'Кого будем искать?',
reply_markup= addgenderToSearchKeyboard(message))
            bot.register_next_step_handler(msg, changegenderToSearch)
            #msg = bot.send_message(message.from_user.id, 'Введи свой город')
            #bot.register_next_step_handler(msg, addCity)

        def changegenderToSearch(message):
            connection = psycopg2.connect(
                host=host,
```

```
            user=user,
            password=password,
            dbname=db_name)
        # database = db_name)
        with connection.cursor() as cursor:
            if message.text == "Парней":
                cursor.execute(
                    "UPDATE users SET searchgender = 'M' WHERE telegramid =
%s;", (message.from_user.id,) )
            elif message.text == "Девушек":
                cursor.execute(
                    "UPDATE users SET searchgender = 'Ж' WHERE telegramid =
%s;", (message.from_user.id,) )
            elif message.text == "Всех":
                cursor.execute(
                    "UPDATE users SET searchgender = 'All' WHERE telegramid =
%s;", (message.from_user.id,) )
        connection.commit()
        connection.close()

        msg = bot.send_message(message.from_user.id, 'Введи свой город')
        bot.register_next_step_handler(msg, changeCity)

    def changeCity(message):
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            dbname=db_name)
        # database = db_name)
        with connection.cursor() as cursor:
            cursor.execute(
                "UPDATE users SET city = %s WHERE telegramid = %s;",
((message.text), (message.from_user.id,)))
        connection.commit()
        connection.close()
        # bot.send_message(message.from_user.id, 'Введи свой возраст')
        ###bot.send_message(message.from_user.id, 'Ты из ' + str(anketa.city))
        msg = bot.send_message(message.from_user.id, 'Напиши что будет в
описании твоего профиля')
        bot.register_next_step_handler(msg, changeDescription)

    def changeDescription(message):
        connection = psycopg2.connect(
```

```
                host=host,
                user=user,
                password=password,
                dbname=db_name)
            # database = db_name)
            with connection.cursor() as cursor:
                cursor.execute(
                    "UPDATE users SET description = %s WHERE telegramid = %s;",
((message.text), (message.from_user.id,)))
            connection.commit()
            connection.close()
            # bot.send_message(message.from_user.id, 'Введи свой возраст')
            #bot.send_message(message.from_user.id, 'Описание: ' +
str(anketa.descripition))
            msg = bot.send_message(message.from_user.id, 'Добавь пожалуйста твое
фото')
            bot.register_next_step_handler(msg, changeMedia)

        # def showInfoNotMedia(message):
        #     infoNotMedia =  anketa.name + ', ' + anketa.age + ', ' + anketa.city + '\n' +
anketa.descripition
        #     return infoNotMedia
        #     #bot.send_message(message.from_user.id, text = infoNotMedia , text=
downloaded.file)
        #     #bot.send_message()

        def showMedia(message):
            pass

        # def addMedia(message):
        #     file_info = bot.get_file(message)
        #     file =
requests.get('https://api.telegram.org/file/bot{0}/{1}'.format(API_TOKEN,
file_info.file_path))
        #     file_info = bot.get_file(anketa.file_id)
        #     file =
requests.get('https://api.telegram.org/file/bot{0}/{1}'.format(API_TOKEN,
file_info.file_path))
        #    # bot.send_message(message.from_user.id, 'Введи свой возраст')
        #    # bot.send_message(message.from_user.id, 'СТОЯТЬ')
        #     anketa.vidpics = message.text
        #     showMedia(message)
        #     showInfoNotMedia(message)
```

```
    #    ####ДОБАВИТЬ KEYBOARD ДЛЯ МЕНЮ ВЫБОРА ЧТО ДЕЛАТЬ ДАЛЬШЕ
    #     #bot.send_message(message.from_user.id, 'Описание: ' + str(anketa.descripition))
    #     #msg = bot.send_message(message.from_user.id, 'Хорошо, теперь загрузи пожалуйста свои фото или видео')
    #     #bot.register_next_step_handler(msg, addAge)

    @bot.message_handler(content_types='photo')
    def changeMedia(message):
        #bot.send_message(message.from_user.id, text = 'Спасибо за фотографию!')

        raw = message.photo[2].file_id
        path = raw + ".jpg"
        file_info = bot.get_file(raw)
        downloaded_file = bot.download_file(file_info.file_path)
        with open(path, 'wb') as new_file:
            new_file.write(downloaded_file)

        global fotka
        fotka = downloaded_file

        bot.send_message(message.from_user.id, text='Так выглядит твой профиль:')

        mainkeyboard = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
        button_searchProfiles = types.KeyboardButton(text="Поиск других профилей")
        button_changeProfile = types.KeyboardButton(text="Изменить профиль")
        button_quitProfile = types.KeyboardButton(text="Выключить профиль")
        mainkeyboard.add(button_searchProfiles, button_changeProfile, button_quitProfile)

        bot.send_photo(message.from_user.id, downloaded_file, caption = showInfoNotMedia(message), reply_markup=mainkeyboard)
    ################REPEAT END
```

```
if __name__ == '__main__':
    print('main function && create keyboard')
    bot.infinity_polling()
```

userinterface.py

```
import anketa
import main
import userinterface
from main import bot
from anketa import callback
# global callback
#from main import callback
from telebot import types


# userinterface.callback = main.callback
global currentKeyboard

def addInitialKeyboard(message):
    keyboard = types.ReplyKeyboardMarkup(row_width=1, resize_keyboard=True)
    button_initialCreateProfile = types.KeyboardButton(text='Создать профиль')
    # global callback
    anketa.callback = 'Создать профиль'
    #callback = 'Создать профиль'
    keyboard.add(button_initialCreateProfile)

    global currentKeyboard
    currentKeyboard = keyboard

    bot.send_message(message.chat.id, text='Нажми кнопку "Создать профиль"',
reply_markup=keyboard)
    return anketa.callback

def addGenderKeyboard(message):
    keyboard = types.ReplyKeyboardMarkup(row_width=2, resize_keyboard=True)
    button_genderMale = types.KeyboardButton(text="Парень")
    button_genderFemale = types.KeyboardButton(text="Девушка")
    keyboard.add(button_genderMale, button_genderFemale)

    global currentKeyboard
    currentKeyboard = keyboard
```

```
    bot.send_message(message.chat.id, text = 'Выбери свой пол', reply_markup=keyboard)
    return keyboard

def addgenderToSearchKeyboard(message):
    keyboard = types.ReplyKeyboardMarkup(row_width=3, resize_keyboard=True)
    button_searchMale = types.KeyboardButton(text="Парней")
    button_searchFemale = types.KeyboardButton(text="Девушек")
    button_searchAll = types.KeyboardButton(text= "Всех")
    keyboard.add(button_searchMale, button_searchFemale, button_searchAll)

    global currentKeyboard
    currentKeyboard = keyboard

    bot.send_message(message.chat.id, text = 'Кого будем искать?', reply_markup=keyboard)
    return keyboard
# def createProfile:
# button_initialCreateProfile
```

Config.py

```
host = "127.0.0.1"
user = "postgres"
password = "123456"
db_name = "bot_users"
#port = 5432
```