



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Информационный портал «Наставник»

Студент ИУ6-52Б
(Группа)

(Подпись, дата) С.В. Астахов
(И.О. Фамилия)

Руководитель курсовой работы

(Подпись, дата) _____
(И.О. Фамилия)

Консультант

(Подпись, дата) _____
(И.О. Фамилия)

2021 г.

Задание на курсовую работу

Реферат

Расчетно-пояснительная записка 36 страниц, 3 части, 18 рисунков, 3 таблицы, 5 источников, 1 приложение.

ВЕБ-ПРИЛОЖЕНИЕ, ЧАТ-БОТ, НАСТАВНИЧЕСТВО, DJANGO.

Объектом разработки является информационный портал «Наставник».

Цель работы – проектирование и реализация информационного портала, используемого для поиска и общения студентов младших курсов с наставниками, являющимися учащимися старших курсов той же кафедры.

В результате работы был спроектирован и реализован программный комплекс, состоящий из веб-приложения, чат-бота в мессенджере telegram и консоли администратора, позволяющий выбрать оптимальный для пользователя формат взаимодействия с информационным порталом и организовать модерацию аккаунтов. Также при разработке использовался структурный контроль и оценочное тестирование, в том числе, автоматизированное.

Пользователями разработанного приложения могут быть студенты ВУЗа, на базе которого развернуто приложение, если эти студенты заинтересованы в поиске наставника, либо же сами хотят быть наставниками.

Содержание

Введение.....	6
1. Анализ требований и уточнение спецификаций.....	7
1.1. Анализ задания и выбор технологии, языка и среды разработки.....	7
1.2. Выбор модели жизненного цикла программного обеспечения.....	8
1.3. Разработка инфологической модели базы данных.....	8
2. Проектирование структуры и компонентов программного продукта.....	11
2.1. Разработка структуры и компонентов веб-приложения.....	11
2.1.1. Разработка структурной схемы веб-приложения.....	11
2.1.2. Разработка интерфейса пользователя.....	12
2.1.2.1. Построение диаграммы состояний интерфейса.....	12
2.1.2.2. Разработка форм интерфейса.....	15
2.2. Разработка структуры и компонентов чат-бота.....	20
2.2.1. Разработка структурной схемы чат-бота.....	20
2.2.2. Разработка интерфейса чат-бота.....	20
2.2.2.1. Разработка схемы иерархии меню.....	21
2.2.2.2. Разработка форм интерфейса.....	23
3. Выбор стратегии тестирования и разработка тестов.....	26
3.1. Тестирование чат-бота.....	26
3.1.1. Оценочное тестирование.....	26
3.1.2. Структурный контроль.....	28
3.2. Тестирование веб-приложения.....	31
Заключение.....	35
Литература.....	36
ПРИЛОЖЕНИЕ А. Техническое задание	37

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ТЗ – техническое задание.

БД – база данных.

СУБД – система управления базами данных.

Фреймворк — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Model-View-Controller (MVC, «Модель-Представление-Контроллер») — схема разделения данных приложения, и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

Модель (Model) — компонент схемы MVC, который предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) — компонент схемы MVC, который отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) — компонент схемы MVC, который интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Аватар — графическое представление пользователя, двумерное изображение (иконка) или трёхмерная модель.

Модератор — пользователь на общественных сетевых ресурсах , имеющий более узкие права, чем администратор, но более широкие права, чем обычные пользователи. В отличие от администратора, чаще всего следит за соблюдением правил ресурса.

Чат-бот — программа, осуществляющая взаимодействие с пользователем через интерфейс интернет-чата.

Inline клавиатура (InlineKeyboard) — клавиатура привязанная к сообщению, использующая обратный вызов, вместо отправки сообщения с обыкновенной клавиатуры.

Введение

Курсовая работа посвящена проектированию и разработке информационного портала поиска наставников по учебе, представляющего собой программный комплекс из веб-приложения, ориентированного на работу на ПК и ноутбуках (т.е. в веб-браузерах с большим размером и альбомной ориентацией окна), чат-бота, предназначенного для взаимодействия через мобильное приложение “Telegram” и консоли администратора, необходимой для управления правами модераторов. Разрабатываемый информационный портал может быть использован студентами, заинтересованными в участии в программе наставничества. Этот портал позволяет пользователю выбрать наставника, являющегося студентом более старших курсов той же кафедры, что и пользователь, самому стать наставником, управлять системой “заявок в друзья” и осуществлять поиск верифицированных наставников (т.е. студентов, чья личность подтверждена модератором на основании документов).

В ходе выполнения работы в качестве аналогов были рассмотрены чат-боты, предназначенные для взаимодействия и обучения сотрудников внутри той или иной компании, а также сервисы знакомств. Найти детальное описание более функционально близких аналогов в свободном доступе не удалось. Актуальность разработки заключается в том, что в рассмотренных аналогах адаптация под использования в целях реализации программы наставничества либо невозможно в принципе, либо крайне затруднительна и малоэффективна.

1. Анализ требований и уточнение спецификаций

1.1. Анализ задания и выбор технологии, языка и среды разработки

Для разработки чат-бота был выбран смешанный подход программирования, сочетающий событийное и модульное программирование. Событийное программирование лежит в основе абсолютного большинства современных чат-ботов, так как оно позволяет реагировать на приход любого нового сообщения или обратного вызова, как на событие, для которого вызывается соответствующий содержанию обработчик. Модульный подход же позволяет разбить обработчики событий на модули в соответствии с иерархией меню, что упрощает проектирование и работу с кодом программы.

Для разработки веб-приложения выбрана парадигма Model-View-Controller, являющаяся наиболее популярной и устоявшейся в сфере веб-программирования и сочетающая в себе элементы объектного подхода для работы с БД и объектами предметной области со структурным подходом при работе с интерфейсом.

Для разработки как чат-бота, так и веб-приложения был использован язык Python. В случае чат-бота Python был выбран как наиболее популярный язык для написания чат-ботов, имеющий соответствующие библиотеки (в случае этой работы - aiogram) и документацию к ним [1]. В случае веб-приложения был выбран фреймворк Django, как широкоиспользуемый MVC-фреймворк с качественной документацией, позволяющий писать код быстро и компактно в силу динамической типизации и других особенностей языка Python [2].

В качестве редактора исходного кода был выбран Visual Studio Code, позволяющий установить необходимые расширения как для работы с Python, так и для разработки представлений в веб-приложении, т.е. работы с HTML, CSS, JavaScript. Вместе с необходимыми расширениями VS Code позволяет использовать функции автодополнения

кода, автоформатирования, подсказки параметров функций, обнаружение ошибок синтаксиса.

В качестве СУБД был выбран PostgreSQL за ее скорость, свободу распространения, качество документации и наличие библиотек для интеграции с Python [3].

1.2. Выбор модели жизненного цикла программного обеспечения

В связи с объемом и уровнем технической сложности проекта, в качестве модели жизненного цикла была выбрана инкрементальная модель (поэтапная модель с промежуточным контролем). Разработка программного обеспечения ведется итерациями с циклами обратной связи между этапами. Межэтапные корректировки позволяют учитывать реально существующее взаимовлияние результатов разработки на различных этапах, время жизни каждого из этапов растягивается на весь период разработки.

В начале работы над проектом определяются все основные требования к системе, подразделяются на более и менее важные. После чего выполняется разработка системы по принципу приращений, так, чтобы разработчик мог использовать данные, полученные в ходе разработки ПО.

Такая модель позволяет прийти к компромиссу между гибкостью проекта, что свойственно спиральной модели, и низкими накладными расходами на уточнение и изменение требований, что свойственно каскадной модели.

1.3. Разработка инфологической модели базы данных

Разрабатываемая система работает с учетными данными пользователей, а также с их “заявками в друзья” к потенциальным кураторам. Кроме того, необходимо хранить информацию о количестве кафедр на факультете для контроля вводимых при регистрации значений. Также система работает с фото пользователей, что, в связи со спецификой фреймворка Django, требует отдельной таблицы с информацией о месте хранения фото. На рисунке 1 приведена инфологическая модель базы данных.

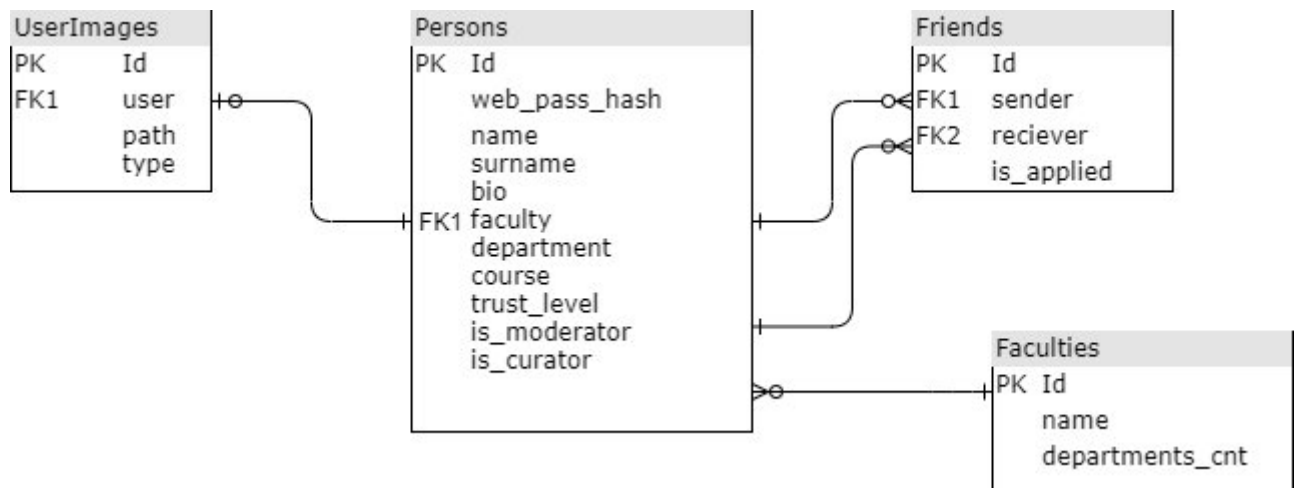


Рисунок 1 - Инфологическая модель базы данных

В базе данных присутствуют следующие таблицы и поля:

- 1) Persons - учетные данные пользователей:
 - id - идентификатор пользователя;
 - web_pass_hash - хэши паролей пользователей для web-приложения;
 - name - имя пользователя;
 - surname - фамилия пользователя;
 - bio - поле “о себе”;
 - faculty - идентификатор факультета;
 - department - кафедра;
 - course - год обучения;
 - trust_level - статус верификации;
 - is_moderator - флаг модератора;
 - is_curator - флаг куратора;
- 2) Faculties - данные о факультетах:
 - id - идентификатор факультета;
 - name - название факультета;
 - departments_cnt - число кафедр на факультете;
- 3) Friends - таблица “заявок в друзья”:

- id - номер заявки;
 - sender - идентификатор отправителя заявки;
 - reciever - идентификатор получателя заявки;
 - is_applied - флаг одобрения заявки;
- 4) UserImages - служебная информация о картинках:
- id - номер картинки;
 - user - идентификатор хозяина картинки;
 - path - путь к картинке;
 - type - тип картинки;

2. Проектирование структуры и компонентов программного продукта

2.1. Разработка структуры и компонентов веб-приложения

2.1.1. Разработка структурной схемы веб-приложения

Для разработки веб-приложения было решено взять за основу схему MVC, поскольку она является одной из стандартных схем для разработки веб-приложений и на ее использование ориентирован используемый фреймворк Django. Исходя из этого необходимо выявить основные модели предметной области, представления и шаблоны и контроллеры.

Проанализировав предметную область, можно заключить, что основными моделями являются “Пользователь” и “Заявка в друзья” (эта модель описывает, в том числе, и принятые заявки).

Пользователь будет прежде всего взаимодействовать со списками других пользователей (заявок) и редактировать собственный профиль, для чего понадобятся соответствующие шаблоны и контроллеры.

Разработанная структурная схема веб-приложения представлена на рисунке 2.

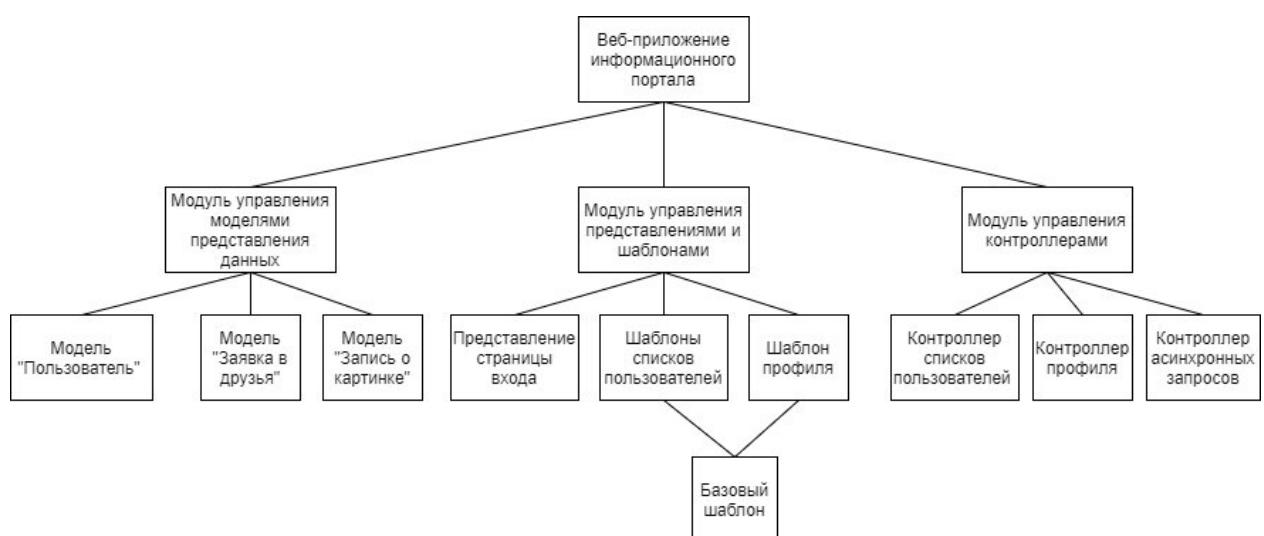


Рисунок 2 - Структурная схема веб-приложения

Кроме моделей “Пользователь” и “Заявка в друзья”, на схеме изображена служебная модель “Запись о картинке”, которая необходима для корректного взаимодействия фреймворка Django с пользовательскими картинками (аватарами и фото документов для верификации аккаунта).

Помимо описанных выше моделей и контроллеров, на схеме отображены: представление страницы входа и базовый шаблон, от которого будет наследоваться боковое и верхнее меню. Кроме того, показан контроллер для работы с асинхронными запросами, с помощью которых осуществляется работа с “заявками в друзья”.

2.1.2. Разработка интерфейса пользователя

В ходе работы с веб-приложением пользователю необходимо просматривать различные типы “заявок в друзья”, а также просматривать и редактировать профиль.

Исходя из частотного принципа проектирования интерфейсов, вынесем ссылки на страницы с различными типами заявок в боковое меню. Исходя из сложившихся в области проектирования веб-интерфейсов шаблонов, ссылки на профиль пользователя и домашнюю страницу расположены в верхнем меню, как и кнопки смены языка и выхода. Кроме того, необходимо предусмотреть необходимые сообщения при некорректном заполнении форм.

2.1.2.1. Построение диаграммы состояний интерфейса

Исходя из требований к пользовательскому интерфейсу и проектных решений, принятых выше, составим диаграмму состояний интерфейса (рисунок 3). На основе этой диаграммы в дальнейшем был спроектирован пользовательский интерфейс. На диаграмме приняты следующие обозначения:

C1 - Авторизация с корректными параметрами

C2 - Попытка авторизации с некорректными параметрами

C3 - Нажатие на название приложения

C4 - Наведение мыши на зону меню

C5 - Нажатие на ссылку "Аккаунт"

- C6 - Нажатие на ссылку "Безопасный поиск"
- C7 - Нажатие на ссылку "Поиск"
- C8 - Нажатие на ссылку "Входящие"
- C9 - Нажатие на ссылку "Исходящие"
- C10 - Нажатие на ссылку "Друзья"
- C11 - Нажатие кнопки "Верифицировать"
- C12 - Нажатие кнопки "Сменить аватар"
- C13 - Нажатие кнопки "Редактировать информацию"
- C14 - Отправка корректно заполненной формы
- C15 - Попытка отправки некорректно заполненной формы
- C16 - Нажатие на ссылку "Выйти"
- C17 - Закрытие страницы веб-приложения
- C18 - Нажатие кнопки "Удалить аватар"
- C19 - Нажатие кнопки "Вкл./выкл. режим куратора"

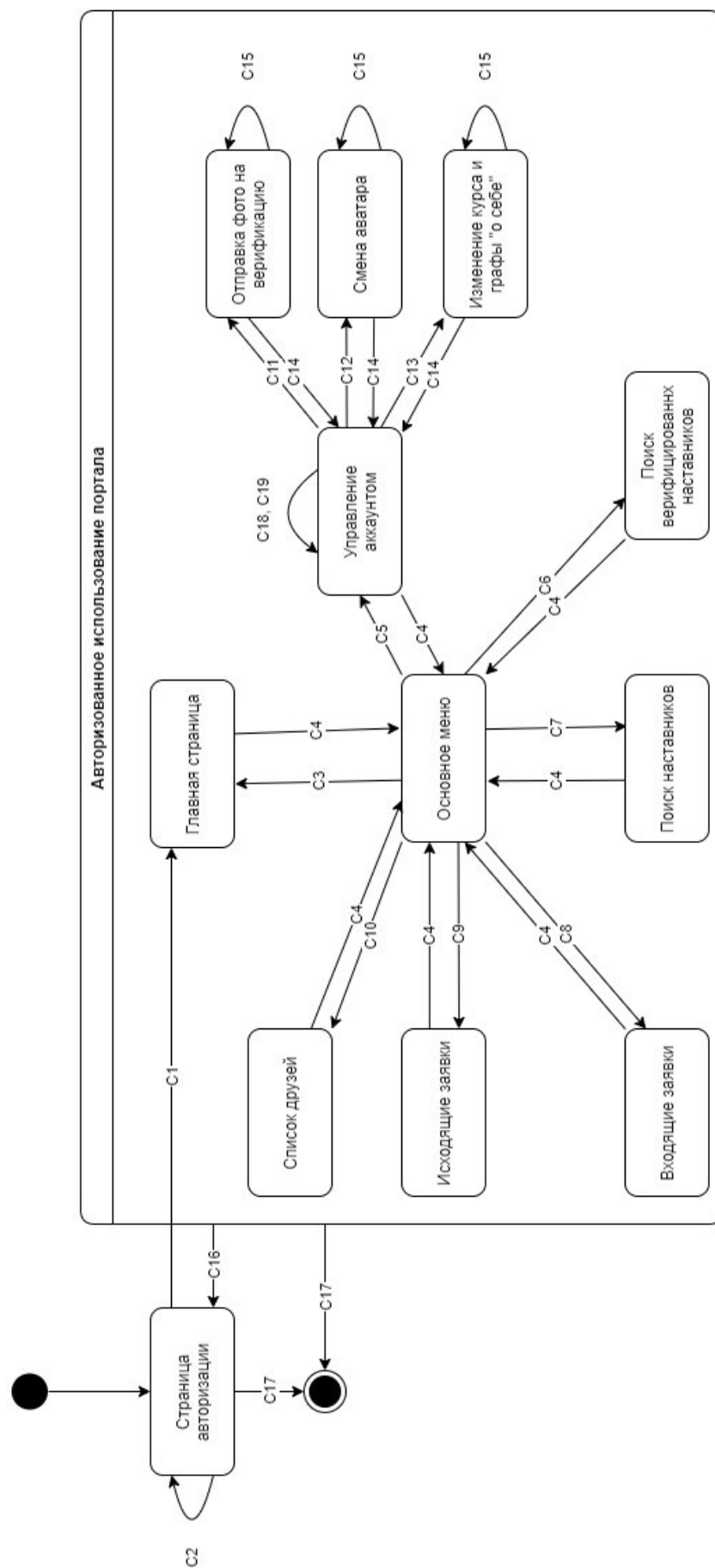


Рисунок 3 - Диаграмма состояний веб-интерфейса

2.1.2.2. Разработка форм интерфейса

На основе полученной выше диаграммы состояний интерфейса был спроектирован веб-интерфейс. Для улучшения визуального восприятия интерфейса был использован фреймворк Bootstrap.

Страница входа позволяет аутентифицироваться на основе данных, полученных с помощью чат-бота, также можно изменить язык приложения. Эта страница представлена на рисунке 4.

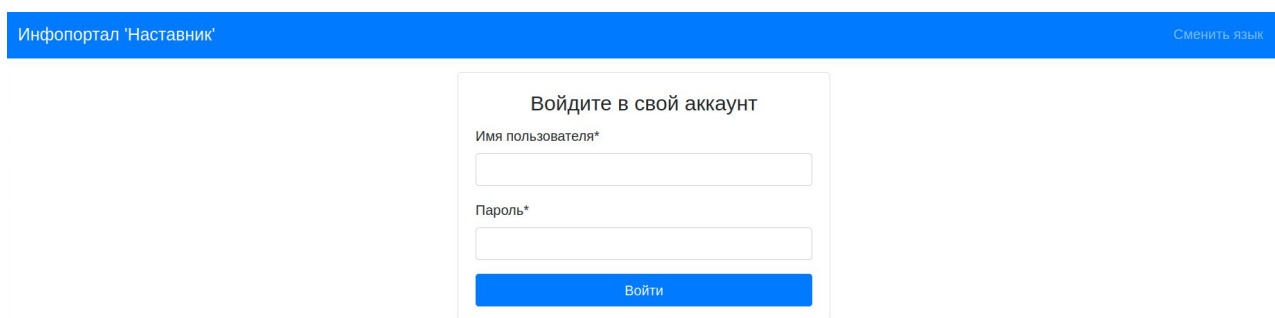


Рисунок 4 - Форма входа

Главная страница представляет собой короткое приветственное сообщение и меню для дальнейшей навигации (рисунок 5).

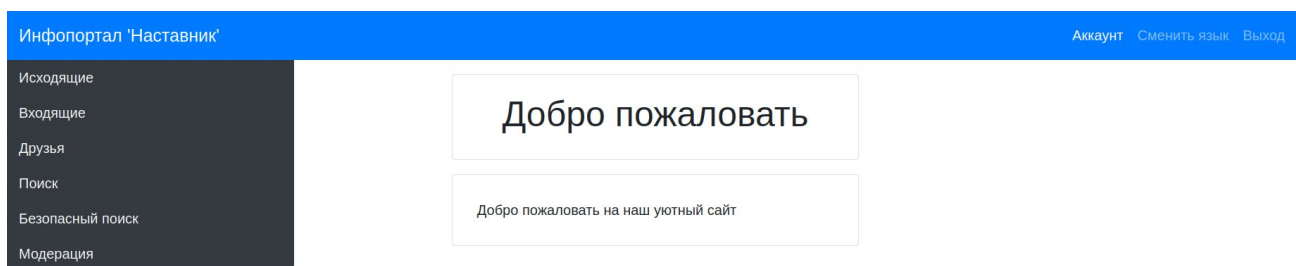


Рисунок 5 - Главная страница

Страницы различных типов “заявок в друзья” и список “друзей” однотипны, поэтому для примера приведен лишь список исходящих заявок (рисунок 6). Кроме того, на приведенных ниже рисунках в целях увеличения масштаба обрезаны боковое и верхнее меню.

Если пользователь не загрузил аватар, будет проставлен аватар по умолчанию, как показано на рисунке 7.

Исходящие



Даниил Иванов


Кафедра: СГН1
Курс: 4
О себе: биография Д.И. - Lorem ipsum dolor sit amet
Модерация: на проверке

Отменить



Рисунок 6 - Список исходящих “заявок в друзья”

Поиск



Николай Тихомиров

Кафедра: СГН1
Курс: 4
О себе: биография Н.Т. - Lorem ipsum dolor sit amet
Модерация: проверен

Подписаться

Рисунок 7 - Карточка с аватаром по умолчанию

Если в том или ином списке отсутствуют пользователи, при попытке открыть список будет отображено соответствующее сообщение (рисунок 8).

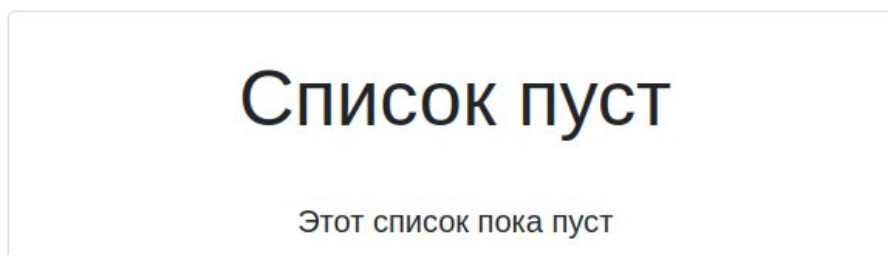


Рисунок 8 - Отображение пустого списка

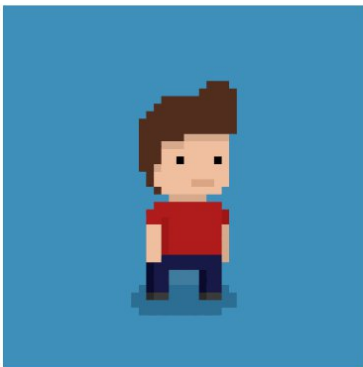
Для верификации аккаунта пользователя модератору необходимо сравнить данные его аккаунта и фотографию с каким-либо документом. Поэтому в списке модерлируемых аккаунтов, кроме данных, стандартных для описанных выше списков, представлено фото документа. Кроме того, карточки пользователей в разделе модерации шире для удобства работы модератора. Страница модерации представлена на рисунке 9.

The screenshot shows a moderation interface. At the top, a header box contains the word "Модерация". Below this, a user card is displayed. The card features a circular profile picture of a man in a white lab coat and a passport icon with a globe and the word "PASSPORT". The user's name "Даниил Иванов" is centered below the images. Further down, the user's details are listed: "ID: 1", "Telegram: @uname1", "Кафедра: СГН1", "Курс: 4", and a bio "О себе: биография Д.И. - Lorem ipsum dolor sit amet". The status "Модерация: на проверке" is shown. At the bottom, there are two checkboxes: "Модератор:" and "Куратор:", both currently unchecked. Below these are two buttons: a blue "Подтвердить" button and a red "Отказать" button.

Рисунок 9 - Страница модерации

Страница управления аккаунтом, изображенная на рисунке 10, отображает информацию о пользователе и позволяет изменить аватар, изменить курс и графу “о себе”, отправить заявку на верификацию аккаунта, удалить аккаунт.

Профиль



Иван Иванов

ID: 421423205
Telegram: @trickster2038

Кафедра: СГН1
Курс: 3
О себе: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat
Модерация: проверен

Модератор: ☒
Куратор: ☐

Изменить режим куратора

Верифицировать

Аватар

Редактировать информацию

Удалить аватар

Удалить профиль

Рисунок 10 - Страница управления аккаунтом

В случае изменения курса или информации о себе, необходимо заполнить соответствующую форму, при этом текстовое поле ограничено по длине текста и не должно быть пустым. Форма представлена на рисунке 11. Если курс был изменен, аккаунт приобретет статус “не проверен”.

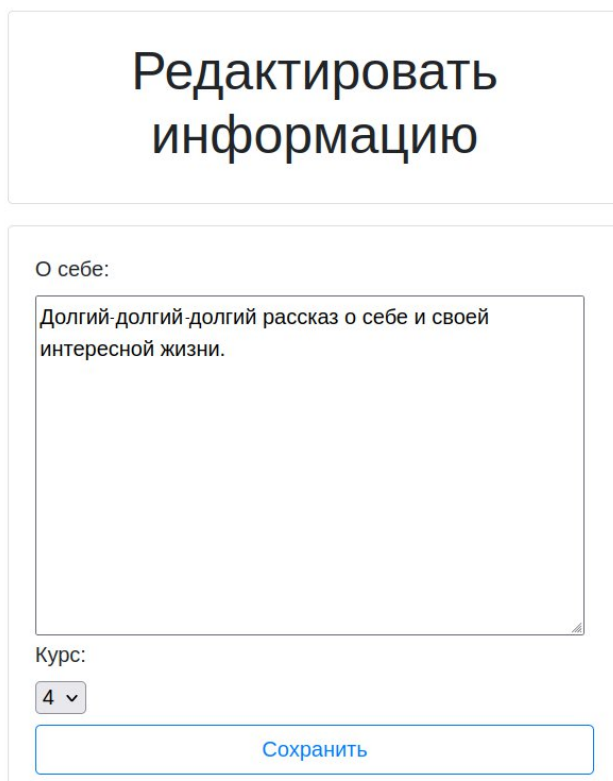


Рисунок 11 - Редактирование курса и графы “о себе”

В случае изменения аватара или отправки заявки на верификацию аккаунта, необходимо прикрепить фото к соответствующей форме, при этом статус верификации аккаунта автоматически изменится. На рисунке 12 показан пример формы изменения аватара (форма подачи заявки на верификацию аналогична).

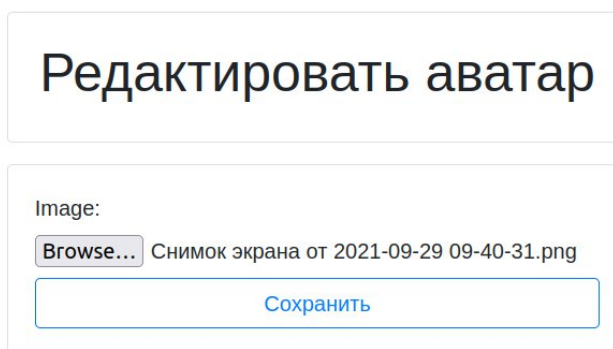


Рисунок 12 - Изменение аватара

2.2. Разработка структуры и компонентов чат-бота

2.2.1. Разработка структурной схемы чат-бота

Для удобства, обособим конфигурационные данные, такие как параметры для подключения к БД и telegram-токен бота в отдельный файл для удобства настройки бота. Разобьем обработчики событий на модули в соответствии с выполняемыми задачами. Все компоненты, очевидно, будут взаимодействовать с БД, вынесем логику работы с БД в отдельный модуль. Структурная схема чат-бота показана на рисунке 13.

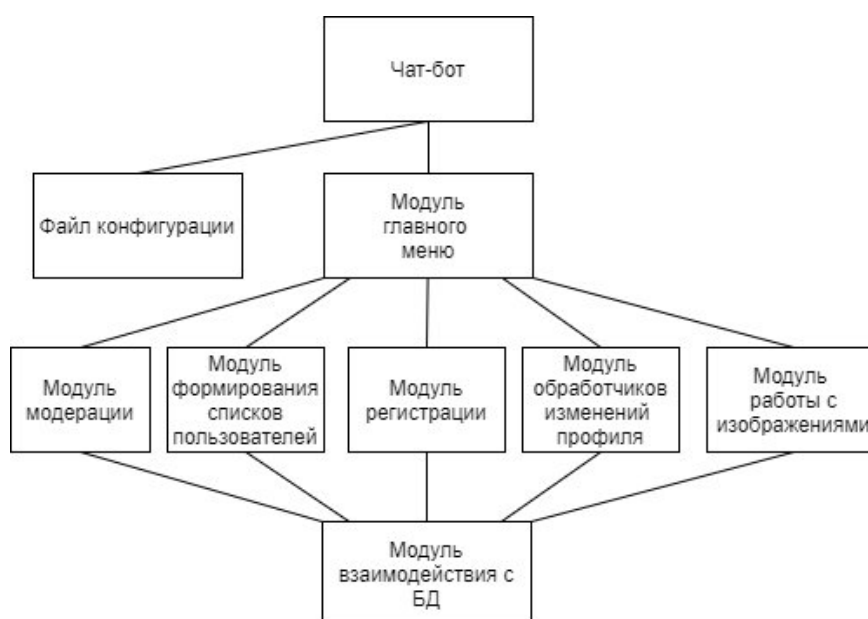


Рисунок 13 - структурная схема чат-бота

Как видно из диаграммы, большинство модулей представляет из себя наборов обработчиков, необходимых для взаимодействия с меню. Таким образом, структурная схема используется как основа для разработки схемы иерархии меню.

2.2.2. Разработка интерфейса чат-бота

Для взаимодействия с пользователем при реализации современных чат-ботов как правило используется меню, если соответствующий мессенджер или интернет-чат предоставляет возможность интеграции такого меню непосредственно в интерфейс чата. В случае мессенджера “Telegram” такая возможность имеется, поэтому разработанный чат-бот имеет тип интерфейса меню [4].

2.2.2.1. Разработка схемы иерархии меню

Исходя из требований ТЗ и анализа предметной области на верхнем уровне иерархии меню выделено четыре пункта:

- профиль;
- регистрация;
- модерация;
- друзья.

На следующем уровне категории “профиль” и “друзья” разворачиваются еще в 7 и 6 подкатегорий соответственно. Такая организация меню позволяет пользователю отображать не слишком большое число категорий на одном уровне иерархии, но при этом не порождать чрезмерное число уровней иерархии. Последний аспект важен, поскольку встроенное в чат меню отображает только пункты одного уровня иерархии одновременно и на любой переход требуется дополнительное нажатие, к тому же большое число уровней иерархии сложно для восприятия.

Разработанная схема иерархии меню представлена на рисунке 14.

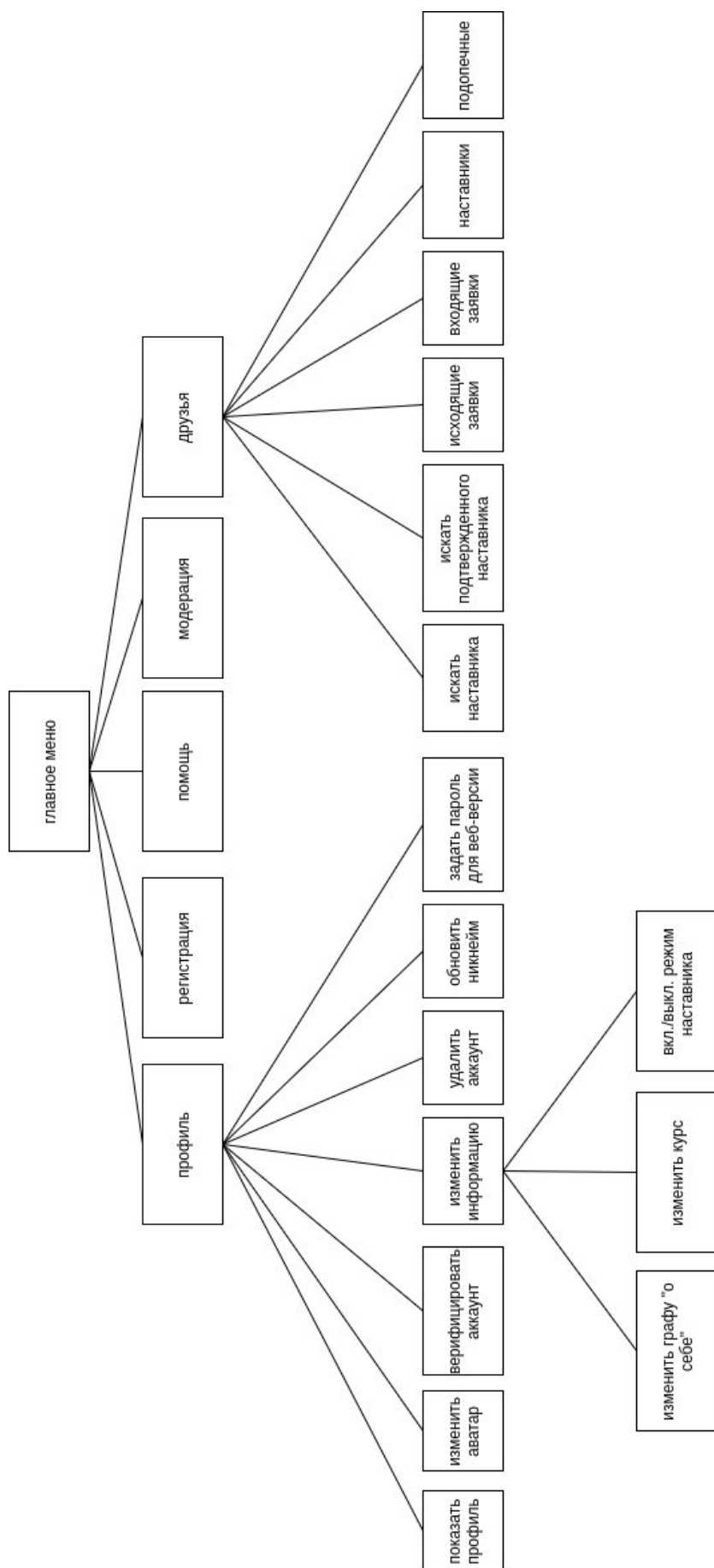


Рисунок 14 - схема иерархии меню

2.2.2.2. Разработка форм интерфейса

Интерфейс реализован на базе предоставляемых Telegram компонентов, а именно inline клавиатур и интерактивного списка команд.

Верхний уровень иерархии меню (рисунок 15) реализован на базе команд Telegram, сопровождаемых интерактивным списком. При этом команда перехода в режим модерации в интересах удобства обычных пользователей в список не включена и должна быть введена команда “/moderate” в текстовом режиме. При попытке несанкционированного доступа к режиму модерации, будет выдано соответствующее сообщение.

Также добавлена команда “/help”, позволяющая вывести краткое руководство по использованию чат-бота и получению учетных данных для работы с веб-приложением.

Кроме того, в связи с тем что, что пользовательский интерфейс использует для сохранения промежуточных состояний конечные автоматы, была добавлена команда “/cancel”. Она необходима для сброса их состояния в случае некорректного поведения или для аварийного выхода из режимов регистрации и редактирования аккаунта.

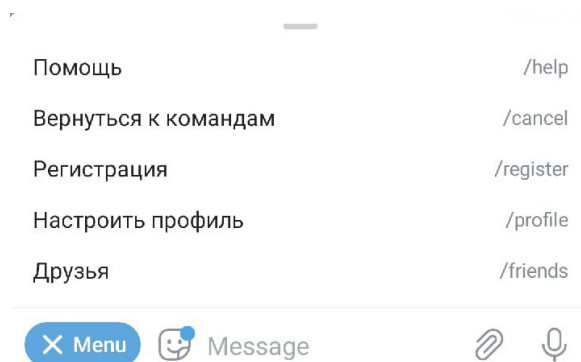


Рисунок 15 - Реализация верхнего уровня меню

Нижележащие уровни иерархии меню реализованы на базе inline клавиатур, а их содержание полностью соответствует схеме иерархии меню. В качестве примера, на рисунке 16 показано подменю “Друзья”.

Списки пользователей разных типов, выводимые при взаимодействии с подменю “Друзья” реализованы как наборы сообщений с привязанной inline-клавиатурой для

принятие или отклонения “заявок в друзья” и тому подобных действий. Пример такого списка приведен на рисунке 17.

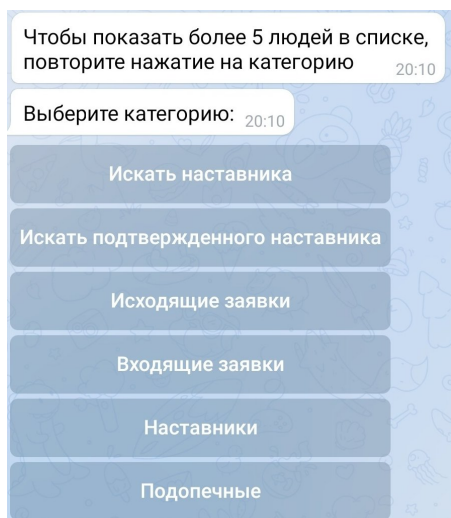


Рисунок 16 - Подменю “Друзья”

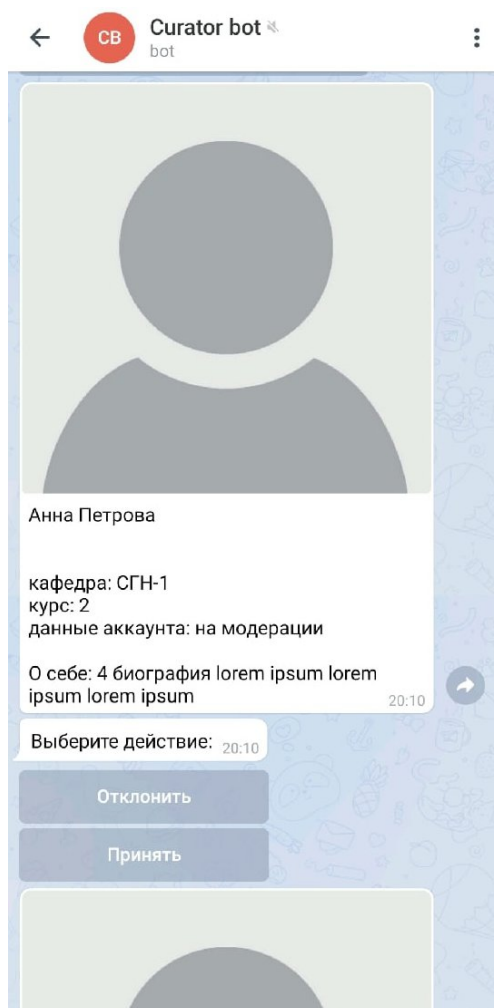


Рисунок 17 - Список друзей

Информация о текущем пользователе представлена карточкой, аналогичной карточкам в списке друзей.

Ввод данных при регистрации и редактировании профиля осуществляется в форме диалога, как показано на рисунке 18.



Рисунок 18 - Ввод данных при регистрации пользователя

3. Выбор стратегии тестирования и разработка тестов

3.1. Тестирование чат-бота

В виду того, что бизнес-логика чат-бота главным образом реализована непосредственно внутри обработчиков событий, было решено использовать в ходе разработки структурный контроль и оценочное тестирование.

3.1.1. Оценочное тестирование

В ходе анализа предметной области был выявлен ряд пользовательских историй, на которых основаны тесты, представленные в таблице 1. Перед началом тестирования, в тестовой БД устанавливается исходное состояние с помощью специального сценария на Python.

Таблица 1 - оценочное тестирование

Номер теста	Описание теста	Ожидаемый результат	Результат	Вывод
1	Зайти в список входящих заявок и принять заявку.	Карточка соответствующего пользователя переместилась в списке друзей.	Карточка соответствующего пользователя переместилась в списке друзей.	Тест прошел успешно.
2	Зайти в список исходящих заявок и удалить заявку из списка.	При повторном открытии списка соответствующая заявка не появилась.	При повторном открытии списка соответствующая заявка не появилась.	Тест прошел успешно.

Продолжение таблицы 1

3	Изменить курс обучения.	В профиле изменился курс и статус модерации аккаунта принял значение “не проверен”.	В профиле изменился курс и статус модерации аккаунта принял значение “не проверен”.	Тест прошел успешно.
4	Изменить аватар.	Аватар изменен и статус модерации принял значение “не проверен”.	Аватар изменен и статус модерации принял значение “не проверен”.	Тест прошел успешно.
5	Изменить верификационное фото.	Аккаунт изменил статус модерации на “на проверке”.	Аккаунт изменил статус модерации на “на проверке”.	Тест прошел успешно.
6	Удалить аккаунт.	Аккаунт недоступен.	Аккаунт недоступен.	Тест прошел успешно.
7	Зарегистрироваться с произвольными параметрами (имя и фамилия - кириллицей).	В профиле отобразилась информация, введенная при регистрации.	В профиле отобразилась информация, введенная при регистрации.	Тест прошел успешно.

В результате оценочного тестирования удалось убедиться в корректности работы чат-бота.

3.1.2. Структурный контроль

В ходе разработки был использован метод структурного контроля, позволяющий определить типовые ошибки, часто встречающиеся в коде. Стандартный список вопросов для структурного контроля был изменен в соответствии со спецификой разрабатываемого продукта, например, убрана большая часть вопросов, касающихся арифметических вычислений [5]. На промежуточных этапах разработки были выявлены и устранены такие ошибки, как некорректный выход из цикла и “наложение” некорректного типа на нетипизированные переменные. Результаты заключительного структурного контроля приведены в таблице 2.

Таблица 2 - Структурный контроль

Вопрос	Результаты структурного контроля	Вывод
Все ли переменные инициализированы?	Да, иначе среда разработки подсветила бы ошибку.	Все переменные инициализированы.
Присутствуют ли переменные со сходными именами?	Нет.	Переменные со сходными именами отсутствуют.
Использованы ли нетипизированные переменные, открытые массивы, динамическая память? Если да, то соответствуют ли типы переменных при "наложении" формата?	Использованы нетипизированные и динамические массивы. Типы хранимых значений соответствуют ожидаемым.	Обращение к нетипизированным переменными не вызывает ошибок.

Продолжение таблицы 2

Не выходят ли индексы за границы массивов?	Индексы не выходят за границы массивов, так как они или перебираются в специальных итерационных циклах или их структура соответствует запросам к БД.	Индексы не выходят за границы массивов.
Будут ли корректно завершены циклы?	Да, все циклы - итерационные циклы по значениям массивов.	Циклы будут завершены корректно.
Существуют ли циклы, которые не будут выполняться из-за нарушения условия входа? Корректно ли продолжатся вычисления?	Да. Работа программы продолжится корректно, будут выведены соответствующие сообщения.	Ситуация невыполнения тела цикла обрабатывается корректно.
Существуют ли поисковые циклы? Корректно ли отрабатываются ситуации "элемент найден" и "элемент не найден"?	Поисковых циклов нет.	Поисковых циклов нет.

Продолжение таблицы 2

Соответствуют ли списки параметров и аргументов подпрограмм по порядку, типу, единицам измерения?	Да.	Вызовы подпрограмм происходят корректно.
Не изменяет ли подпрограмма аргументов, которые не должны изменяться?	Не изменяет.	Подпрограммы не изменяют аргументов, которые не должны изменяться.
Не происходит ли нарушения области действия глобальных и локальных переменных с одинаковыми именами?	Нарушения области действия глобальных и локальных переменных не происходит.	Нарушения области действия глобальных и локальных переменных не происходит.

По результатам структурного контроля можно сделать вывод, что в заключительной версии проекта типовые ошибки кодирования были устранены.

3.2. Тестирование веб-приложения

Так, как разработанное веб-приложение не предполагает ввода данных значительного объема или сложности, для его тестирования была выбрана стратегия оценочного тестирования.

Тесты были автоматизированы с помощью библиотек Pytest и Selenium и проверяют корректность как бизнес-логики приложения, так и корректность отображения интерфейса. Тестовая БД возвращается в исходное состояние перед каждым тестом.

Описание и результаты тестов представлены в таблице 3.

Таблица 3 - оценочное тестирование

Название теста	Описание теста	Ожидаемый результат	Полученный результат	Вывод
test_login	Аутентификация с корректными параметрами.	Переход на главную страницу, появление кнопки “Выйти”.	Переход на главную страницу, появление кнопки “Выйти”.	Тест прошел успешно.
test_login_negative	Аутентификация с некорректными параметрами.	Выдача сообщения о некорректных данных.	Выдача сообщения о некорректных данных.	Тест прошел успешно.

Продолжение таблицы 3

test_accept_delete_friend	Принятие “заявки в друзья”, последующее удаление из друзей.	Появление карточки в списке “друзей” после добавления и ее исчезновение после удаления.	Появление карточки в списке “друзей” после добавления и ее исчезновение после удаления.	Тест прошел успешно.
test_delete_or_subscribe_cards	Параметризованный тест, проверяющий исчезновение карточек из входящих заявок при отклонении, из исходящих - при удалении, из поиска - при отправке заявки.	Исчезновение карточки из заданного списка после совершения действия.	Исчезновение карточки из заданного списка после совершения действия.	Тест прошел успешно.
test_moderate	Параметризованный тест, проверяющий изменение статуса карточки после модерации.	Статус карточки изменился на заданный в параметрах.	Статус карточки изменился на заданный в параметрах.	Тест прошел успешно.

Продолжение таблицы 3

test_switch_mode	Проверка переключения в режим куратора и обратно.	Флажок “куратор” изменяет состояние.	Флажок “куратор” изменяет состояние.	Тест прошел успешно.
test_photo_upload	Параметризованный тест. Проверка корректной работы при отправке аватара или фото для верификации.	Изменение статуса аккаунта на указанный в параметрах.	Изменение статуса аккаунта на указанный в параметрах.	Тест прошел успешно.
test_photo_upload_negative	Проверка обработки пустой формы при попытке отправить фото.	Выдача соответствующего сообщения, невыполнение перехода к странице профиля.	Выдача соответствующего сообщения, невыполнение перехода к странице профиля.	Тест прошел успешно.

Продолжение таблицы 3

test_edit_account	Параметризованный тест. Проверяет изменение данных в профиле.	Данные, указанные в параметрах появляются в профиле.	Данные, указанные в параметрах появляются в профиле.	Тест прошел успешно.
test_edit_account_negative	Тест проверяет обработку пустой формы при попытке изменить данные пользователя.	Выдача соответствующего сообщения, невыполнение перехода к странице профиля.	Выдача соответствующего сообщения, невыполнение перехода к странице профиля.	Тест прошел успешно.

В результате оценочного тестирования удалось убедиться в корректности работы веб-приложения.

Заключение

В результате выполнения курсовой работы была спроектирована система, состоящая из веб-приложения на основе фреймворка Django, чат-бота и консоли администратора, полностью удовлетворяющая всем требованиям технического задания.

Разработанная система предоставляет возможности создания и изменения профиля, поиска наставников и модерации аккаунтов.

Система может быть расширена в последующих версиях. В качестве усовершенствования рассматривается добавление системы рейтинга пользователей, введение системы жалоб на пользователей и увеличение объема информации, хранимой профилем.

В ходе разработки системы был приобретён опыт проектирования программных продуктов, работы с системой контроля версий, работы с БД, фреймворком Django, библиотеками Pytest и Selenium, необходимыми для автоматизированного тестирования веб-приложений, а также получены навыки составления технической документации.

Проект расположен в репозитории по адресу:
<https://bmstu.codes/sergey.astahov/cooursework5>.

Автоматизированные тесты расположены в репозитории по адресу:
https://bmstu.codes/sergey.astahov/cw5_tests.

Литература

1. Aiogram Documentation [Электронный ресурс]. - URL: <https://docs.aiogram.dev/en/latest/> (дата обращения: 25.09.2021).
2. DjangoProject [Электронный ресурс]. - URL: <https://www.djangoproject.com/> (дата обращения: 10.10.2021).
3. PostgreSQL 14.1 Documentation [Электронный ресурс]. - URL: <https://www.postgresql.org/docs/14/index.html> (дата обращения: 15.10.2021).
4. Иванова Г.С., Ничушкина Т.Н., Пугачёв Е.К., Самарёв Р.С., Фетисов М.В. – Методические указания по выполнению курсовой работы по дисциплине «Технология разработки программных систем»: Электронное учебное издание. – МГТУ им. Н.Э. Баумана, 2019. – 41 с.
5. Иванова Г.С. – Технология программирования: учебник / Г.С. Иванова. – 3-е изд., стер. – М. : КНОРУС, 2016. – 334 с. – (Бакалавриат).

ПРИЛОЖЕНИЕ А

Техническое задание

Листов 8