Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Петрозаводский государственный университет»

Институт математики и информационных технологий

кафедра прикладная математика и кибернетика

Отчёт по учебному курсу ”Верификация программного обеспечения”

09.03.04 - программная инженерия

Выполнил:
студент
Мельников И.Е.

Руководитель
к.ф-м.н, доцент
Кулаков К.А

Петрозаводск — 2023

Оглавление

Глава 1

Объект тестирования

1.1 Описание приложения

Веб-приложение "Vesuvio Restaurant"разработано для управления бронированием столов в ресторане. Приложение предоставляет пользователям возможность регистрации, входа в систему, просмотра истории бронирования, а также бронирования столов на определенную дату.

- Backend: Python
- Backend фрэймворк: Flask
- Frontend: HTML, CSS
- Frontend фрэймворк: Bootstrap
- База данных: PostgreSQL

1.2 Функции приложения

- **Регистрация и вход:** Пользователи могут зарегистрироваться, создав учетную запись, или войти в систему, если у них уже есть аккаунт.
- **Просмотр дашборда:** После входа в систему пользователи видят персональный дашборд, на котором отображается информация о зарезервированных столах и предоставляется возможность забронировать новый стол.

- **Бронирование столов:** Пользователи могут выбрать дату и забронировать столы на выбранную дату. При выборе столов отображается схема ресторана с указанием доступных и зарезервированных столов.
- **Подтверждение бронирования:** После выбора столов пользователи получают подтверждение бронирования с возможностью сохранения брони в базе данных.
- **Уведомление по электронной почте:** Система отправляет уведомления по электронной почте пользователю после успешного бронирования столов.
- **Просмотр истории бронирования:** Пользователи могут просматривать историю своих предыдущих бронирований.
- **Административные функции:** Администраторы могут просматривать общую информацию о заявках на бронирование и удалять нежелательные запросы.

1.3 Архитектура приложения

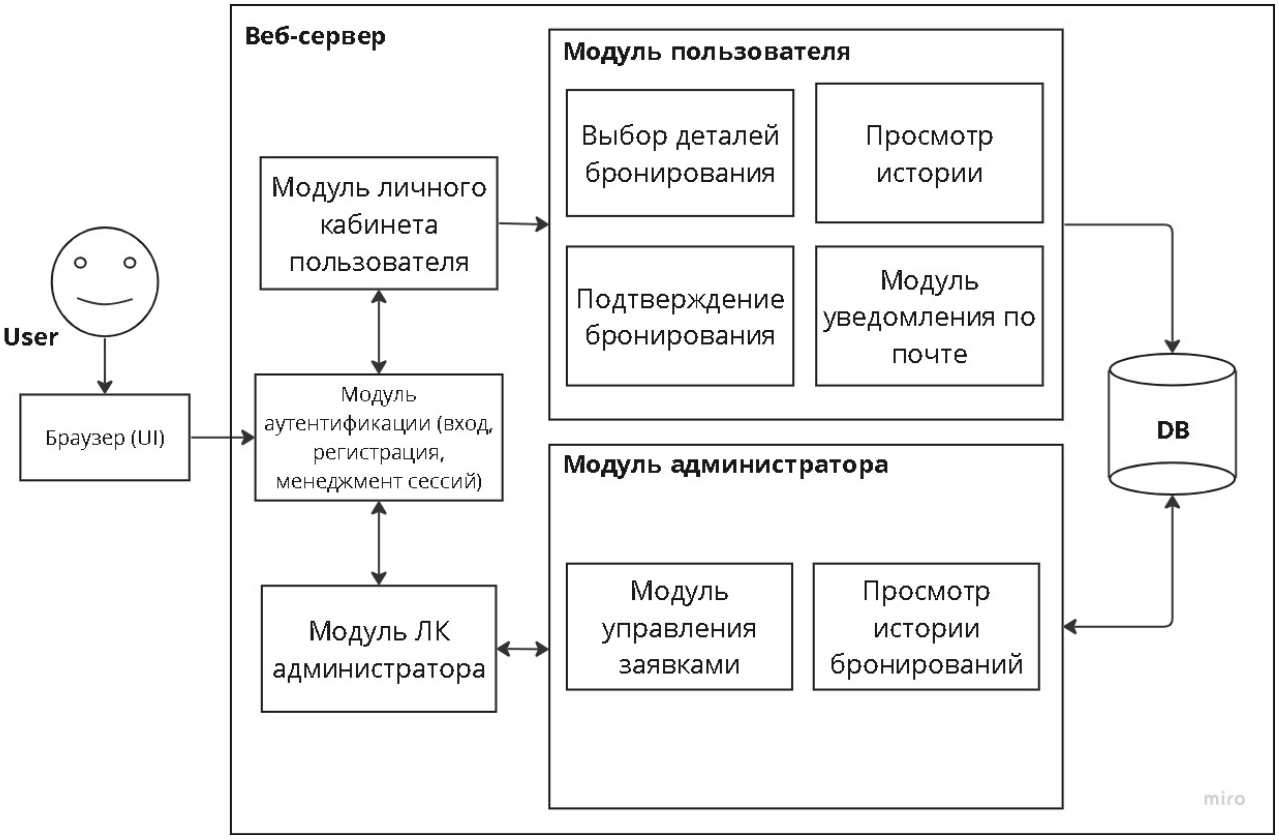

Рис. 1.1: Архитектура приложения

1.4 Описание модулей

- auth_routes.py Модуль для обработки маршрутов, связанных с аутентификацией и авторизацией пользователей.
- **reservation_routes.py** Модуль, отвечающий за обработку запросов, связанных с бронированием столов.
- **admin_routes.py** Модуль, содержащий обработчики запросов, предназначенные для административных функций.
- **reservation_history**$_routes.py$, .
- **email_templates.py** Модуль, содержащий шаблоны электронных писем, используемых для уведомлений.
- **db_utils.py** Утилиты для взаимодействия с базой данных, включая создание, чтение, обновление и удаление данных.
- **user_utils.py** Утилиты для работы с пользователями, такие как создание нового пользователя, проверка прав доступа и другие.
- **models.py** Описание моделей данных, которые хранятся в базе данных, например, модель пользователя, модель бронирования и т.д.
- **main.py** Основной файл, запускающий веб-приложение, настраивающий маршруты и другие параметры.
- **forms**/ Директория, в которой могут храниться формы для ввода данных, например, форма входа, форма регистрации и другие.
- **templates**/ Директория, содержащая HTML-шаблоны для отображения страниц приложения, включая страницы входа, регистрации, панели управления и другие.

1.5 Описание методов и классов

1.5.1 auth_routes.py:

- **Класс AuthRoutes**
  - `login()`: Метод для обработки запросов на вход в систему.

- `register()`: Метод для обработки запросов на регистрацию нового пользователя.
- `logout()`: Метод для выхода из системы.

1.5.2 reservation_routes.py:

- **Класс ReservationRoutes**
  - `select_date()`: Метод для отображения страницы выбора даты бронирования.
  - `select_tables()`: Метод для обработки запросов на выбор столов для бронирования.
  - `process_tables()`: Метод для обработки запросов на подтверждение бронирования столов.
  - `dashboard()`: Метод для отображения панели пользователя с информацией о забронированных столах.

1.5.3 admin_routes.py:

- **Класс AdminRoutes**
  - `admin_dashboard()`: Метод для отображения панели администратора с обзором пользователей и бронирований.
  - `admin_delete_request()`: Метод для обработки запросов администратора на удаление бронирования.

1.5.4 reservation_history_routes.py:

- **Класс ReservationHistoryRoutes**
  - `reservation_history()`: Метод для отображения страницы с историей бронирования.

1.5.5 email_templates.py:

- **Класс EmailTemplates**
  - `generate_confirmation_email()`: Метод для генерации электронного письма с подтверждением бронирования.

1.5.6 db_utils.py:

- **Класс DBUtils**
  - `create_user()`: Метод для создания нового пользователя в базе данных.
  - `get_user_by_username()`: Метод для получения пользователя по имени пользователя.
  - `create_reservation()`: Метод для создания новой записи о бронировании в базе данных.
  - `get_reserved_tables()`: Метод для получения списка забронированных столов.

1.5.7 user_utils.py:

- **Класс UserUtils**
  - `is_admin()`: Метод для проверки, является ли пользователь администратором.
  - `has_reservation()`: Метод для проверки, есть ли у пользователя активные бронирования.

1.5.8 models.py:

- **Классы User и Reservation**
  - `User`: Модель данных пользователя с полями, такими как имя пользователя, пароль и др.
  - `Reservation`: Модель данных бронирования стола с полями, такими как дата, номер стола и др.

Глава 2

Стратегия тестирования

2.1 Стратегия блочного тестирования

Unit-тестирование (модульное тестирование) — процесс, позволяющий проверить на корректность единицы исходного кода, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки. Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Для тестирования был выбран встроенный python модуль - unittest, который поддерживает автоматизацию тестов, использование общего кода для настройки и завершения тестов, объединение тестов в группы, а также позволяет отделять тесты от фреймворка для вывода информации.

- **Тестирование модуля Регистрации и Авторизации (`auth_routes.py`):**
  - `login()`:
    - Ввод корректных учетных данных и проверка успешного входа.
    - Ввод некорректного имени пользователя и проверка сообщения об ошибке.
    - Ввод некорректного пароля и проверка сообщения об ошибке.
    - Проверка корректного перенаправления после входа.
  - `register()`:
    - Ввод корректных учетных данных и проверка успешной регистрации.

        * Ввод уже существующего имени пользователя и проверка сообщения об ошибке.
        * Ввод некорректных данных (пустое имя пользователя или пароль) и проверка сообщений об ошибке.
        * Проверка корректного перенаправления после регистрации.
    - `logout()`:
        * Проверка корректного выхода из системы и перенаправления на главную страницу.

- **Тестирование модуля управления бронированием (`reservation_routes.py`):**
    - `select_date()`:
        * Проверка корректного отображения страницы выбора даты.
    - `select_tables()`:
        * Проверка корректного отображения страницы выбора столов.
        * Выбор доступных столов и проверка успешного бронирования.
        * Попытка бронирования уже занятого стола и проверка сообщения об ошибке.
    - `process_tables()`:
        * Проверка корректной обработки запросов на подтверждение бронирования.
    - `dashboard()`:
        * Проверка корректного отображения панели пользователя с информацией о бронированиях.

- **Тестирование модуля администрирования (`admin_routes.py`):**
    - `admin_dashboard()`:
        * Проверка корректного отображения панели администратора с обзором пользователей и бронирований.
    - `admin_delete_request()`:
        * Проверка корректной обработки запросов администратора на удаление бронирования.

- **Тестирование модуля истории бронирования (`reservation_history_routes.py`):**

  - `reservation_history()`:
    * Проверка корректного отображения страницы с историей бронирования.

- **Тестирование модуля шаблонов электронной почты (`email_templates.py`):**
  - `generate_confirmation_email()`:
    * Проверка корректной генерации письма с подтверждением бронирования.

- **Тестирование модуля утилит для работы с базой данных (`db_utils.py`):**
  - Тестирование методов создания пользователя и бронирования, получения данных о пользователе и забронированных столах.

- **Тестирование модуля утилит для работы с пользователями (`user_utils.py`):**
  - Тестирование методов проверки администратора и наличия активных бронирований у пользователя.

- **Тестирование модуля моделей (`models.py`):**
  - Тестирование моделей данных `User` и `Reservation` на корректность создания и сохранения записей в базе данных.

2.2 Стратегия интеграционного тестирования

Интеграционное тестирование — одна из фаз тестирования программного обеспечения, при которой отдельные программные модули объединяются и тестируются в группе. Обычно интеграционное тестирование проводится после модульного тестирования и предшествует системному тестированию.

Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования. Инструменты - JUnit, DbUnit.

- **Интеграция `auth_routes.py` с `db_utils.py`:**
  - Тестирование регистрации нового пользователя и проверка корректного сохранения данных в базе данных.

  - Тестирование аутентификации пользователя и проверка корректного доступа к данным из базы данных.

- **Интеграция** `reservation_routes.py` **с** `auth_routes.py`**:**
  - Тестирование возможности пользователя выбрать дату и стол после успешной аутентификации.
  - Проверка корректного отображения данных пользователя после бронирования.

- **Интеграция** `reservation_routes.py` **с** `reservation_history_routes.py`**:**
  - Тестирование добавления нового бронирования и проверка его отображения в истории бронирования.
  - Тестирование корректной обработки запросов пользователя к истории бронирования после бронирования стола.

- **Интеграция** `admin_routes.py` **с** `auth_routes.py`**:**
  - Тестирование доступа администратора к данным пользователей после аутентификации.
  - Проверка корректного обновления данных администратором (например, удаление бронирований).

- **Интеграция** `reservation_routes.py` **с** `admin_routes.py`**:**
  - Тестирование возможности администратора просматривать и управлять бронированиями пользователей.

- **Интеграция** `reservation_routes.py` **с** `email.py`**:**
  - Тестирование отправки уведомлений об успешном бронировании пользователям.

- **Интеграция** `reservation_routes.py` **с** `db_utils.py`**,** `user_utils.py`**, и** `models.py`**:**
  - Тестирование корректной работы всех утилит и моделей в процессе бронирования.

- **Интеграция** `reservation_routes.py` **с** `select_date.py` **и** `select_tables.py`**:**
  - Тестирование перехода пользователя от выбора даты к выбору стола.

- Проверка корректного сохранения выбранной даты и стола перед бронированием.

- **Интеграция** `admin_routes.py` **с** `admin_delete_request.py`**:**
  - Тестирование возможности администратора просматривать и удалять запросы на удаление бронирования.

- **Интеграция** `reservation_routes.py` **с** `confirmation.py`**:**
  - Тестирование отправки подтверждения бронирования пользователю после успешного выбора стола.

- **Интеграция** `reservation_history_routes.py` **с** `db_utils.py` **и** `models.py`**:**
  - Тестирование корректного отображения истории бронирования с использованием утилит и моделей базы данных.

- **Интеграция** `select_date.py` **и** `select_tables.py` **с** `reservation_routes.py`**:**
  - Тестирование передачи данных о выбранной дате и столе в процессе бронирования.

2.3 Стратегия аттестационного тестирования

Аттестационное тестирование - процесс тестирования пользователем, являющийся по сути первичным обнаружением ошибок. Методы ручного тестирования достаточно эффективны с точки зрения нахождения ошибок, так что один или несколько из них должны использоваться в каждом программном проекте.

- **Регистрация и вход в систему:**
  - **Тестирование регистрации:**
    - Проверка успешной регистрации пользователя с корректными данными.
    - Проверка отображения сообщения об ошибке при попытке регистрации с недопустимыми данными.
  - **Тестирование входа в систему:**
    - Проверка успешного входа пользователя с корректными учетными данными.

    * Проверка отображения сообщения об ошибке при входе с недопустимыми данными.

- **Просмотр дашборда:**
  - Тестирование корректного отображения дашборда пользователя после входа в систему.
  - Проверка наличия необходимых элементов на дашборде (например, кнопок для бронирования, просмотра истории и др.).
- **Бронирование стола:**
  - Полный цикл бронирования от выбора даты до подтверждения.
  - Тестирование отображения ошибки при попытке бронирования, если все столы заняты.
- **Подтверждение бронирования:**
  - Тестирование подтверждения бронирования.
  - Проверка уведомления пользователя о подтвержденном ронировании.
  - Тестирование отображения подтвержденных бронирований в истории пользователя.
- **Панель администратора:**
  - Тестирование удаления бронирования из панели администратора.
  - Тестирование корректного отображения истории бронирования пользователей.
- **История бронирования:**
  - Тестирование корректного отображения истории бронирования пользователя.
- **Отправка уведомлений:**
  - Проверка отправки уведомлений пользователю после успешного бронирования.

2.4 Стратегия нагрузочного тестирования

Нагрузочным называется тестирование, направленное на проверку способности программы эффективно работать при пиковых или очень высоких нагрузках, связанных с большим количеством запросов пользователей.

- **Тестируемый функционал:** Оценка производительности системы при большом числе запросов и пользователей.
- **Сценарии тестирования:**
  - Тестирование одновременного бронирования столов разными пользователями.
  - Тестирование авторизации большого числа пользователей одновременно.
  - Тестирование работы системы при большом объеме данных в истории бронирований.
  - Тестирование обработки множественных запросов на подтверждение бронирования.
- **Инструменты:** Apache JMeter. Мощный инструмент с открытым исходным кодом для проведения нагрузочного тестирования. Поддерживает разнообразные протоколы, включая HTTP, HTTPS, JDBC, FTP и многие другие.
- **Метрики производительности:**
  - Время отклика системы при различных нагрузках.
  - Пропускная способность системы.
  - Загрузка сервера и базы данных.
  - Общее время выполнения типичных сценариев использования.
- **Процедура тестирования:**
  - Постепенное увеличение числа одновременных пользователей.
  - Запуск длительных тестов для оценки стабильности системы при продолжительной нагрузке.
  - Анализ результатов и выявление узких мест системы.

2.5 Критерии прохождения тестирования

2.5.1 Блочное тестирование

- Все модули успешно прошли юнит-тестирование, проверяющее их базовую функциональность.
- Успешное создание, регистрация и аутентификация пользователей.
- Корректное сохранение и извлечение данных из базы данных.
- Верное отображение информации на пользовательском интерфейсе.
- Эффективная обработка запросов, связанных с бронированием столов.

2.5.2 Интеграционное тестирование

- Согласованное взаимодействие между модулем регистрации и авторизации, модулем бронирования, и панелями администратора и пользователя.
- Корректная обработка данных при передаче от клиентской части к серверной и обратно.
- Успешная интеграция с внешними сервисами для отправки уведомлений по электронной почте.
- Обработка запросов от различных типов пользователей (администратор, обычный пользователь) без конфликтов.

2.5.3 Аттестационное тестирование

- Функциональность "Просмотр дашборда"отображает актуальные данные о состоянии ресторана.
- Успешное подтверждение бронирования и отправка уведомлений об успешном бронировании.
- Правильная обработка запросов на удаление резервации администратором.

2.5.4 Нагрузочное тестирование

- Система успешно справляется с пиковой нагрузкой, соответствующей заявленной максимальной загрузке.
- Время отклика системы остается в приемлемых пределах даже при повышенной нагрузке.
- Пропускная способность системы соответствует заявленным требованиям при обработке большого числа запросов.
- Все выявленные узкие места системы были устранены, и система поддерживает заданный уровень производительности.

2.6 Условия возобновления и приостановки выполнения тестов

2.6.1 Условия приостановки выполнения тестов

- Обнаружение критических ошибок, которые могут повлиять на стабильность работы системы.
- Необходимость внесения существенных изменений в функциональность приложения или его архитектуру.
- Внешние факторы (сбои)

2.6.2 Условия возобновления выполнения тестов

- Устранение критических ошибок и проверка стабильности работы системы.
- Завершение внесения существенных изменений с успешным регрессионным тестированием.

Глава 3

Детальный план тестирования

3.1 Блочное тестирования

3.1.1 Тестирование модуля регистрации и авторизации (auth_routes.py

login()

- **B1:** Ввод корректных учетных данных и проверка успешного входа.
  - Входные данные:

    Пользователь: "username": "valid_username "password": "valid_password
  - Шаги: Попытка входа с корректными данными.
  - Результат: Успешный вход в систему.
- **B2:** Ввод некорректного имени пользователя и проверка сообщения об ошибке.
  - Входные данные:

    Пользователь: "username": "invalid_username "password": "valid_password
  - Шаги: Попытка входа с некорректными данными.
  - Результат: Сообщение об ошибке о некорректном имени пользователя.
- **B3:** Ввод некорректного пароля и проверка сообщения об ошибке.
  - Входные данные:

    Пользователь: "username": "valid_username "password": "invalid_password
  - Шаги: Попытка входа с некорректными данными.
  - Результат: Сообщение об ошибке о некорректном пароле.

- **B4:** Проверка корректного перенаправления после входа.
  - Входные данные:

    Пользователь: "username": "valid_username "password": "valid_password
  - Шаги: Попытка входа с корректными данными.
  - Результат: Перенаправление на целевую страницу (например, дашборд).

register()

- **B5:** Ввод корректных учетных данных и проверка успешной регистрации.
  - Входные данные:

    Пользователь: "username": "new_username "password": "new_password
  - Шаги: Попытка регистрации с корректными данными.
  - Результат: Успешная регистрация в системе.
- **B6:** Ввод уже существующего имени пользователя и проверка сообщения об ошибке.
  - Входные данные:

    Пользователь: "username": "existing_username "password": "new_password
  - Шаги: Попытка регистрации с существующим именем пользователя.
  - Результат: Сообщение об ошибке о существующем имени пользователя.
- **B7:** Ввод некорректных данных (пустое имя пользователя или пароль) и проверка сообщений об ошибке.
  - Входные данные:

    Пользователь: "username": , "password": "new_password
  - Шаги: Попытка регистрации с некорректными данными.
  - Результат: Сообщения об ошибке о некорректных данных.
- **B8:** Проверка корректного перенаправления после регистрации.
  - Входные данные:

    Пользователь: "username": "new$_u$*sername*"*password*" : "*new$_p$assword*
  - Шаги: Попытка регистрации с корректными данными.
  - Результат: Перенаправление на целевую страницу (например, дашборд).

logout()

- **B9:** Проверка корректного выхода из системы и перенаправления на главную страницу.
  - Входные данные:

    Пользователь, авторизованный в системе.
  - Шаги: Выход из системы.
  - Результат: Корректный выход из системы и перенаправление на главную страницу.

3.1.2 Тестирование Модуля Управления Бронированием (reservation_routes.py)

select_date()

- **B10:** Проверка корректного отображения страницы выбора даты.
  - Входные данные: Пользователь, авторизованный в системе.
  - Шаги: Переход на страницу выбора даты.
  - Результат: Корректное отображение страницы выбора даты.

select_tables()

- **B11:** Проверка корректного отображения страницы выбора столов.
  - Входные данные: Пользователь, авторизованный в системе.
  - Шаги: Переход на страницу выбора столов.
  - Результат: Корректное отображение страницы выбора столов.
- **B12:** Выбор доступных столов и проверка успешного бронирования.
  - Входные данные: Пользователь, авторизованный в системе, доступные столы.
  - Шаги: Выбор доступных столов и отправка запроса на бронирование.
  - Результат: Успешное бронирование столов.
- **B13:** Попытка бронирования уже занятого стола и проверка сообщения об ошибке.

- Входные данные: Пользователь, авторизованный в системе, занятый стол.
- Шаги: Попытка бронирования занятого стола.
- Результат: Сообщение об ошибке о занятости стола.

process_tables()

- **B14:** Проверка корректной обработки запросов на подтверждение бронирования.
  - Входные данные: Пользователь, авторизованный в системе, с запросом на подтверждение бронирования.
  - Шаги: Отправка запроса на подтверждение бронирования.
  - Результат: Корректная обработка запроса и изменение статуса бронирования.

dashboard()

- **B15:** Проверка корректного отображения панели пользователя с информацией о бронированиях.
  - Входные данные: Пользователь, авторизованный в системе, с активными бронированиями.
  - Шаги: Переход на панель пользователя.
  - Результат: Корректное отображение панели пользователя с информацией о бронированиях.

3.1.3 Тестирование модуля администрирования (admin_routes.py)

admin_dashboard()

- **B16:** Проверка корректного отображения панели администратора с обзором пользователей и бронированиями.
  - Входные данные: Администратор, авторизованный в системе.
  - Шаги: Переход на панель администратора.
  - Результат: Корректное отображение панели администратора с обзором пользователей и бронированиями.
- **B17:** Проверка отсутствия доступа к панели администратора для обычного пользователя.

- Входные данные: Пользователь, авторизованный в системе.
- Шаги: Попытка перехода на панель администратора.
- Результат: Отсутствие доступа и корректное сообщение об ошибке.

admin_delete_request()

- **B17:** Проверка корректной обработки запросов администратора на удаление бронирования.
  - Входные данные: Администратор, авторизованный в системе, с запросом на удаление бронирования.
  - Шаги: Отправка запроса на удаление бронирования.
  - Результат: Корректная обработка запроса и удаление бронирования из системы.
- **B18:** Попытка удаления бронирования обычным пользователем.
  - Входные данные: Пользователь, авторизованный в системе, с запросом на удаление бронирования.
  - Шаги: Отправка запроса на удаление бронирования.
  - Результат: Отсутствие доступа и корректное сообщение об ошибке.

3.1.4 Тестирование модуля истории бронирования (reservation_history

reservation_history()

- **B19:** Проверка корректного отображения страницы с историей бронирования.
  - Входные данные: Пользователь, авторизованный в системе.
  - Шаги: Переход на страницу истории бронирования.
  - Результат: Корректное отображение страницы с историей бронирования пользователя.
- **B20:** Попытка доступа к истории бронирования неавторизованным пользователем.
  - Входные данные: Пользователь, неавторизованный в системе.
  - Шаги: Попытка перехода на страницу истории бронирования.
  - Результат: Отсутствие доступа и корректное сообщение об ошибке.

3.1.5 Тестирование модуля шаблонов электронной почты (email_templ

generate_confirmation_email()

- **B21:** Проверка корректной генерации письма с подтверждением бронирования.
  - Входные данные: Данные успешного бронирования (дата, стол, пользователя и пр.).
  - Шаги: Вызов функции генерации письма с подтверждением.
  - Результат: Сгенерированное письмо соответствует ожидаемому формату и содержанию.
- **B22:** Попытка генерации письма без необходимых данных.
  - Входные данные: Неполные данные бронирования.
  - Шаги: Попытка вызова функции генерации письма.
  - Результат: Генерация завершается ошибкой и возвращается соответствующее сообщение.

3.1.6 Тестирование модуля утилит для работы с базой данных (db_utils.py)

create_user() и create_reservation()

- **B23:** Проверка корректного создания пользователя и бронирования в базе данных.
  - Входные данные: Данные нового пользователя и бронирования.
  - Шаги:
    1. Вызов метода create_user() с новыми данными пользователя.
    2. Вызов метода create_reservation() с данными нового бронирования.
  - Результат: Записи пользователя и бронирования успешно добавлены в базу данных.
- **B24:** Получение данных о пользователе и забронированных столах.
  - Входные данные: Имя пользователя.
  - Шаги:

1. Вызов метода get_user_data() с именем пользователя.
2. Вызов метода get_user_reservations() с именем пользователя.

– Результат: Получены корректные данные о пользователе и забронированных столах.

- **B25:** Попытка создания пользователя с неполными данными.
  - Входные данные: Неполные данные нового пользователя.
  - Шаги: Вызов метода create_user() с неполными данными.
  - Результат: Создание пользователя завершается ошибкой и возвращается соответствующее сообщение.
- **B26:** Получение данных о несуществующем пользователе.
  - Входные данные: Не существующее имя пользователя.
  - Шаги: Попытка вызова метода get_user_data() и get_user_reservations().
  - Результат: Возвращается сообщение об ошибке отсутствия данных.

3.1.7 Тестирование модуля утилит для работы с пользователями (user_utils.py)

is_admin() и has_active_reservations()

- **B27:** Проверка корректного определения администратора.
  - Входные данные: Имя пользователя с правами администратора.
  - Шаги: Вызов метода is_admin() с именем администратора.
  - Результат: Метод возвращает корректный результат - пользователь является администратором.
- **B28:** Проверка отсутствия прав администратора.
  - Входные данные: Имя обычного пользователя.
  - Шаги: Вызов метода is_admin() с именем обычного пользователя.
  - Результат: Метод возвращает корректный результат - пользователь не является администратором.

- **B29:** Проверка наличия активных бронирований у пользователя.
  - Входные данные: Имя пользователя с активными бронированиями.
  - Шаги: Вызов метода has_active_reservations() с именем пользователя.
  - Результат: Метод возвращает корректный результат - у пользователя есть активные бронирования.
- **B30:** Проверка отсутствия активных бронирований у пользователя.
  - Входные данные: Имя пользователя без активных бронирований.
  - Шаги: Вызов метода has_active_reservations() с именем пользователя.
  - Результат: Метод возвращает корректный результат - у пользователя нет активных бронирований.

3.1.8 Тестирование модуля моделей (models.py)

Тестирование Модели User

- **B31:** Проверка корректного создания и сохранения записи пользователя.
  - Входные данные: username, email, password.
  - Шаги: Создание экземпляра модели User и сохранение в базу данных.
  - Результат: Запись пользователя корректно создана и сохранена.
- **B32:** Проверка уникальности имени пользователя.
  - Входные данные: username, email, password.
  - Шаги: Создание двух экземпляров модели User с одинаковыми именами и сохранение в базу данных.
  - Результат: Запись пользователя не сохранена из-за нарушения уникальности имени пользователя.

Тестирование модели Reservation

- **B33:** Проверка корректного создания и сохранения записи бронирования.
  - Входные данные: user_id, selected_date, selected_tables.
  - Шаги: Создание экземпляра модели Reservation и сохранение в базу данных.

- Результат: Запись бронирования корректно создана и сохранена.

- **B34:** Проверка обязательных полей модели.
  - Входные данные: user_id.
  - Шаги: Создание экземпляра модели Reservation без обязательных полей и попытка сохранения в базу данных.
  - Результат: Запись бронирования не сохранена из-за отсутствия обязательных полей.

3.2 Интеграционное тестирования

3.2.1 Тестирование интеграции auth_routes.py с db_utils.py

Тест I1: Тест Регистрации

- **Тип теста:** Позитивный
- **Описание:** Проверка корректной регистрации нового пользователя и сохранения данных в базе данных.
- **Входные данные:**
  - `username = "new_user"`
  - `email = "new_user@example.com"`
  - `password = "secure_password"`
- **Шаги:**
  1. Вызов функции регистрации: `register(username, email, password)`.
  2. Поиск созданной записи пользователя в базе данных: `find_user(username)`.
- **Результат:** Запись нового пользователя с именем "new_user"корректно создана и сохранена в базе данных.

Тест I2: Тест аутентификации

- **Тип теста:** Позитивный

- **Описание:** Проверка корректной аутентификации пользователя и доступа к данным из базы данных.
- **Входные данные:**
  - `username = "auth_user"`
  - `password = "secure_password"`
- **Шаги:**
  1. Регистрация нового пользователя: `register(username, "auth_user@example.com pas`
  2. Вызов функции аутентификации: `login(username, password)`.
  3. Проверка доступа к данным пользователя в базе данных: `get_user_data(username)`.
- **Результат:** Аутентификация прошла успешно, и доступ к данным пользователя с именем "auth_user"в базе данных осуществлен.

Тест I3: Тест аегистрации с ауществующим именем

- **Тип теста:** Негативный
- **Описание:** Проверка обработки попытки регистрации с уже существующим именем пользователя.
- **Входные данные:**
  - `existing_username = "existing_user"`
  - `email = "new_user@example.com"`
  - `password = "secure_password"`
- **Шаги:**
  1. Регистрация пользователя с именем "existing_user": `register(existing_username, "existing_user@example.com password)`.
  2. Повторная попытка регистрации с тем же именем пользователя: `register(existing_u email, password)`.
- **Результат:** Система обнаруживает существующее имя пользователя "existing_user"и возвращает сообщение об ошибке.

3.2.2 Тестирование интеграции reservation_routes.py с auth_routes.py

Тест I4: Тест Выбора Даты и Стола после Аутентификации

- **Тип теста:** Позитивный
- **Описание:** Проверка возможности пользователя выбрать дату и стол после успешной аутентификации.
- **Входные данные:**
  - `username = "auth_user"`
  - `password = "secure_password"`
  - `selected_date = "2023-01-01"`
  - `selected_table = 1`
- **Шаги:**
  1. Аутентификация пользователя: `login(username, password)`.
  2. Выбор даты и стола: `select_date(selected_date)`, `select_tables(selected_table)`
- **Результат:** Пользователь успешно выбирает дату и стол после аутентификации.

Тест I5: Тест отображения данных пользователя после бронирования

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного отображения данных пользователя после бронирования.
- **Входные данные:**
  - `username = "auth_user"`
  - `password = "secure_password"`
  - `selected_date = "2023-01-01"`
  - `selected_table = 1`
- **Шаги:**

1. Аутентификация пользователя: `login(username, password)`.
2. Выбор даты и стола: `select_date(selected_date)`, `select_tables(selected_table)`
3. Просмотр данных пользователя после бронирования: `dashboard()`.

- **Результат:** Отображение корректных данных пользователя, включая последние бронирование, на странице панели пользователя.

Тест I6: Тест бронирования занятого стола

- **Тип теста:** Негативный
- **Описание:** Попытка пользователя забронировать стол, который уже занят.
- **Входные данные:**
  - `username = "auth_user"`
  - `password = "secure_password"`
  - `selected_date = "2023-01-01"`
  - `selected_table_occupied = 1`
- **Шаги:**
  1. Аутентификация пользователя: `login(username, password)`.
  2. Выбор даты и занятого стола: `select_date(selected_date)`, `select_tables(selecte`
- **Результат:** Пользователь получает сообщение об ошибке, указывающее, что выбранный стол уже занят, и бронирование не выполняется.

3.2.3 Тестирование Интеграции reservation_routes.py с reservation_history_routes.py

Тест I7: Тест добавления бронирования в историю

- **Тип теста:** Позитивный
- **Описание:** Проверка добавления нового бронирования и корректного отображения в истории бронирования.
- **Входные данные:**

- username = "auth_user"
- password = "secure_password"
- selected_date = "2023-01-01"
- selected_table_available = 2

- **Шаги:**

  1. Аутентификация пользователя: `login(username, password)`.
  2. Выбор даты и доступного стола: `select_date(selected_date)`, `select_tables(selec`
  3. Подтверждение бронирования: `process_tables(selected_date)`.
  4. Просмотр истории бронирования: `reservation_history()`.

- **Результат:** Бронирование успешно добавлено в историю, и оно корректно отображается при просмотре истории бронирования.

Тест I8: Тест просмотра истории бронирования

- **Тип теста:** Позитивный
- **Описание:** Проверка корректной обработки запросов пользователя к истории бронирования после бронирования стола.
- **Входные данные:**
  - username = "auth_user"
  - password = "secure_password"
- **Шаги:**

  1. Аутентификация пользователя: `login(username, password)`.
  2. Просмотр истории бронирования: `reservation_history()`.

- **Результат:** История бронирования успешно отображается, и пользователь видит свои предыдущие бронирования.

Тест I9: Тест добавления бронирования в историю (Неверные данные)

- **Тип теста:** Негативный
- **Описание:** Проверка обработки попытки добавления бронирования с неверными данными.
- **Входные данные:**
  - `username = "auth_user"`
  - `password = "secure_password"`
  - `selected_date = "2023-01-01"`
  - `selected_table_unavailable = 1` *(стол, который уже занят)*
- **Шаги:**
  1. Аутентификация пользователя: `login(username, password)`.
  2. Выбор даты и недоступного стола: `select_date(selected_date)`, `select_tables(sel`
  3. Попытка подтверждения бронирования: `process_tables(selected_date)`.
- **Результат:** Система должна отобразить сообщение об ошибке, указывающее на то, что выбранный стол уже занят.

Тест I10: Тест просмотра истории бронирования (Неаутентифицированный Пользователь)

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки запроса на просмотр истории бронирования от неаутентифицированного пользователя.
- **Входные данные:**
  - `username = "unauthorized_user"`
  - `password = "invalid_password"`
- **Шаги:**
  1. Попытка просмотра истории бронирования: `reservation_history()`.
- **Результат:** Система должна перенаправить пользователя на страницу входа и отобразить сообщение о необходимости аутентификации.

3.2.4 Тестирование Интеграции admin_routes.py с auth_routes.py

Тест I11: Тест доступа администратора к данным пользователей

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного доступа администратора к данным пользователей.
- **Входные данные:**
  - `admin_username = "admin"`
  - `admin_password = "admin_password"`
- **Шаги:**
  1. Аутентификация администратора: `login(admin_username, admin_password)`.
  2. Просмотр данных пользователей: `admin_dashboard()`.
- **Результат:** Система должна корректно отобразить панель администратора с данными о пользователях.

Тест I12: Тест успешного удаления бронирования администратором

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного удаления бронирования администратором.
- **Входные данные:**
  - `admin_username = "admin"`
  - `admin_password = "admin_password"`
  - `reservation_id_to_delete = 12345`
- **Шаги:**
  1. Аутентификация администратора: `login(admin_username, admin_password)`.
  2. Удаление бронирования: `admin_delete_request(reservation_id_to_delete)`.
- **Результат:** Система должна успешно удалить бронирование с указанным идентификатором.

Тест I13: Тест доступа неадминистратора к данным пользователей

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки попытки доступа неадминистратора к данным пользователей.
- **Входные данные:**
  - `regular_user_username = "regular_user"`
  - `regular_user_password = "regular_password"`
- **Шаги:**
  1. Аутентификация обычного пользователя: `login(regular_user_username, regular_u`
  2. Попытка просмотра данных пользователей: `admin_dashboard()`.
- **Результат:** Система должна перенаправить пользователя на главную страницу с сообщением о нехватке прав доступа.

Тест I14: Тест удаления бронирования обычным пользователем

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки попытки обычного пользователя удалить бронирование.
- **Входные данные:**
  - `regular_user_username = "regular_user"`
  - `regular_user_password = "regular_password"`
  - `reservation_id_to_delete = 12345`
- **Шаги:**
  1. Аутентификация обычного пользователя: `login(regular_user_username, regular_u`
  2. Попытка удаления бронирования: `admin_delete_request(reservation_id_to_delete`
- **Результат:** Система должна отобразить сообщение об ошибке, указывающее на нехватку прав доступа для выполнения операции.

3.2.5 Тестирование интеграции reservation_routes.py с admin_routes.py

Тест I15: Тест просмотра и управления бронированиями Администратором

- **Тип теста:** Позитивный
- **Описание:** Проверка возможности администратора просматривать и управлять бронированиями пользователей.
- **Входные данные:**
  - `admin_username = "admin"`
  - `admin_password = "admin_password"`
- **Шаги:**
  1. Аутентификация администратора: `login(admin_username, admin_password)`.
  2. Просмотр панели администратора: `admin_dashboard()`.
  3. Управление бронированием: `admin_delete_request(reservation_id)` *(произвольный идентификатор бронирования)*.
- **Результат:** Система должна корректно отобразить информацию о бронированиях пользователей и позволить администратору управлять ими.

Тест I16: Тест просмотра и управления бронированиями обычным пользователем

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки попытки обычного пользователя просмотреть и управлять бронированиями.
- **Входные данные:**
  - `regular_user_username = "regular_user"`
  - `regular_user_password = "regular_password"`
- **Шаги:**
  1. Аутентификация обычного пользователя: `login(regular_user_username, regular_u`

2. Попытка просмотра панели администратора: `admin_dashboard()`.
3. Попытка управления бронированием: `admin_delete_request(reservation_id)` *(произвольный идентификатор бронирования).*

- **Результат:** Система должна отобразить сообщение об ошибке, указывающее на нехватку прав доступа для выполнения операции.

3.2.6 Тестирование интеграции reservation_routes.py с email.py

Тест I17: Тест отправки Уведомлений об Успешном Бронировании

- **Тип теста:** Позитивный
- **Описание:** Проверка корректной отправки уведомлений об успешном бронировании пользователям.
- **Входные данные:**
  - `user_username = "test_user"`
  - `user_email = "test_user@example.com"`
  - `reservation_id` *(идентификатор успешного бронирования)*
- **Шаги:**
  1. Выполнение бронирования: `select_date()`, `select_tables()`, `process_tables()`.
  2. Проверка отправки уведомления: `generate_confirmation_email(user_username, user_email, reservation_id)`.
- **Результат:** Система должна успешно отправить уведомление с подтверждением бронирования на адрес пользователя.

Тест I18: Тест Отправки Уведомлений без Бронирования

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки попытки отправки уведомления без предварительного бронирования.
- **Входные данные:**

- `user_username = "test_user"`
- `user_email = "test_user@example.com"`
- `reservation_id` *(не существующий идентификатор бронирования)*

- **Шаги:**

1. Попытка отправки уведомления: `generate_confirmation_email(user_username, user_email, reservation_id)`.

- **Результат:** Система должна обработать сценарий без бронирования и выдать сообщение об ошибке.

3.2.7 Тестирование интеграции reservation_routes.py с db_utils.py, user_utils.py, и models.py

Тест I19: Тест Корректной Работы Утилит и Моделей при Бронировании

- **Тип теста:** Позитивный
- **Описание:** Проверка корректной работы утилит и моделей при успешном бронировании.
- **Входные данные:**
  - `user_username = "test_user"`
  - `user_password = "test_password"`
  - `selected_date = "2023-01-01"`
  - `selected_tables = [1, 2]`
- **Шаги:**
  1. Регистрация пользователя: `auth_routes.register(user_username, user_password)`
  2. Аутентификация пользователя: `auth_routes.login(user_username, user_password`
  3. Выбор даты: `reservation_routes.select_date(selected_date)`.
  4. Выбор столов: `reservation_routes.select_tables(selected_tables)`.
  5. Подтверждение бронирования: `reservation_routes.process_tables()`.
- **Результат:** Система должна корректно использовать утилиты и модели для регистрации, аутентификации и бронирования.

Тест I20: Тест Корректной Работы Утилит и Моделей при Бронировании

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки ошибок утилит и моделей при некорректных данных при бронировании.
- **Входные данные:**
  - `user_username = "test_user"`
  - `user_password = "test_password"`
  - `selected_date = "invalid_date"`
  - `selected_tables = [10, 20]` *(несуществующие столы)*
- **Шаги:**
  1. Регистрация пользователя: `auth_routes.register(user_username, user_password)`
  2. Аутентификация пользователя: `auth_routes.login(user_username, user_password`
  3. Попытка выбора некорректной даты: `reservation_routes.select_date(selected_da`
  4. Попытка выбора несуществующих столов: `reservation_routes.select_tables(sele`
  5. Попытка подтверждения бронирования: `reservation_routes.process_tables()`.
- **Результат:** Система должна корректно обрабатывать ошибки утилит и моделей, выводя сообщения об ошибке.

3.2.8 Тестирование интеграции reservation_routes.py с select_date.py и select_tables.

Тест I21: Тест Перехода от Выбора Даты к Выбору Стола

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного перехода пользователя от выбора даты к выбору стола после успешного выбора даты.
- **Входные данные:**
  - `selected_date = "2023-01-01"`
- **Шаги:**

1. Выбор даты: `select_date.select_date(selected_date)`.
2. Переход к выбору столов: `select_date.next()`.

- **Результат:** Пользователь должен успешно перейти от выбора даты к выбору стола.

Тест I22: Тест Перехода от Выбора Даты к Выбору Стола

- **Тип теста:** Негативный
- **Описание:** Проверка корректного предотвращения перехода пользователя от выбора даты к выбору стола при отсутствии выбранной даты.
- **Входные данные:**
  - `selected_date = null`
- **Шаги:**
  1. Попытка перехода к выбору столов без выбранной даты: `select_date.next()`.
- **Результат:** Система должна предотвратить переход и вывести сообщение об ошибке.

Тест I23: Тест Сохранения Выбранной Даты и Стола перед Бронированием

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного сохранения выбранной даты и стола перед бронированием.
- **Входные данные:**
  - `user_username = "test_user"`
  - `user_password = "test_password"`
  - `selected_date = "2023-01-01"`
  - `selected_tables = [1, 2]`
- **Шаги:**
  1. Регистрация пользователя: `auth_routes.register(user_username, user_password)`
  2. Аутентификация пользователя: `auth_routes.login(user_username, user_password`
  3. Выбор даты: `reservation_routes.select_date(selected_date)`.

4. Выбор столов: `reservation_routes.select_tables(selected_tables)`.
5. Проверка корректного сохранения выбранной даты и столов перед бронированием.

- **Результат:** Система должна успешно сохранить выбранную дату и стол перед бронированием.

Тест I24: Тест Сохранения Выбранной Даты и Стола перед Бронированием

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки ошибок при попытке сохранения выбранной даты и стола перед бронированием с некорректными данными.
- **Входные данные:**
  - `user_username = "test_user"`
  - `user_password = "test_password"`
  - `selected_date = "invalid_date"`
  - `selected_tables = [10, 20]` *(несуществующие столы)*
- **Шаги:**
  1. Регистрация пользователя: `auth_routes.register(user_username, user_password)`
  2. Аутентификация пользователя: `auth_routes.login(user_username, user_password`
  3. Выбор некорректной даты: `reservation_routes.select_date(selected_date)`.
  4. Выбор несуществующих столов: `reservation_routes.select_tables(selected_tabl`
  5. Проверка сообщения об ошибке.
- **Результат:** Система должна корректно обрабатывать ошибки и выводить сообщения об ошибке при некорректных данных.

3.2.9 Тестирование интеграции admin_routes. с admin_delete_request

Тест I25: Тест Просмотра и Удаления Запросов на Удаление Бронирования

- **Тип теста:** Позитивный

- **Описание:** Проверка корректного просмотра и удаления запросов администратором на удаление бронирования.
- **Входные данные:**
  - `admin_username = "admin"`
  - `admin_password = "admin_password"`
  - `reservation_id_to_delete = 123`
- **Шаги:**
  1. Аутентификация администратора: `auth_routes.login(admin_username, admin_pass`
  2. Просмотр запросов на удаление бронирования: `admin_routes.admin_delete_request`
  3. Выбор запроса для удаления: `admin_delete_request.choose_request(reservation_`
  4. Удаление выбранного запроса: `admin_delete_request.delete_request()`.
  5. Проверка корректного удаления запроса на удаление бронирования.
- **Результат:** Система должна успешно позволить администратору просматривать и удалять запросы на удаление бронирования.

Тест I26: Тест Просмотра и Удаления Запросов на Удаление Бронирования

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки ошибок при попытке просмотра и удаления запросов администратором на удаление бронирования с некорректными данными.
- **Входные данные:**
  - `admin_username = "admin"`
  - `admin_password = "admin_password"`
  - `reservation_id_to_delete = -1` *(несуществующий ID)*
- **Шаги:**
  1. Аутентификация администратора: `auth_routes.login(admin_username, admin_pass`
  2. Попытка просмотра запросов с некорректным ID: `admin_delete_request.choose_req`

3. Проверка сообщения об ошибке.

- **Результат:** Система должна корректно обрабатывать ошибки и выводить сообщения об ошибке при попытке просмотра и удаления запросов с некорректными данными.

3.2.10 Тестирование интеграции reservation_routes.py с confirmation.py

Тест I27: Тест Отправки Подтверждения Бронирования

- **Тип теста:** Позитивный
- **Описание:** Проверка корректной отправки подтверждения бронирования пользователю после успешного выбора стола.
- **Входные данные:**
  - `user_email = "user@example.com"`
  - `reservation_details = {"table_number": 5, "date": "2023-12-31"}`
- **Шаги:**
  1. Выбор стола пользователем: `reservation_routes.select_tables(reservation_deta`
  2. Проверка отправки подтверждения бронирования: `confirmation.send_confirmation reservation_details)`.
  3. Проверка успешной отправки подтверждения бронирования на указанный email.
- **Результат:** Система должна успешно отправить подтверждение бронирования пользователю после успешного выбора стола.

Тест I28: Тест Отправки Подтверждения Бронирования

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки ошибок при попытке отправки подтверждения бронирования с некорректными данными.
- **Входные данные:**
  - `user_email = "user@example.com"`

– `reservation_details = {"table_number": -1, "date": "invalid_date"}`

- **Шаги:**

  1. Попытка отправки подтверждения с некорректными данными: `confirmation.send_co reservation_details)`.
  2. Проверка сообщения об ошибке.

- **Результат:** Система должна корректно обрабатывать ошибки и выводить сообщения об ошибке при попытке отправки подтверждения бронирования с некорректными данными.

3.2.11 Тестирование интеграции reservation_history_routes.py с db_ut и models.py

Тест I29: Тест Интеграции Истории Бронирования с Базой Данных

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного отображения истории бронирования с использованием утилит и моделей базы данных.
- **Входные данные:** Отсутствуют (используются тестовые данные из базы данных).
- **Шаги:**

  1. Запрос на получение истории бронирования: `reservation_history_routes.reservati`
  2. Проверка корректного отображения истории бронирования.

- **Результат:** Система должна успешно отобразить историю бронирования с использованием утилит и моделей базы данных.

Тест I30: Тест Интеграции Истории Бронирования с Базой Данных

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки ошибок при попытке запроса истории бронирования с использованием утилит и моделей базы данных.
- **Входные данные:** Отсутствуют (используются некорректные тестовые данные).

- **Шаги:**
  1. Попытка запроса истории бронирования с некорректными данными: `reservation_his`
  2. Проверка сообщения об ошибке.
- **Результат:** Система должна корректно обрабатывать ошибки и выводить сообщения об ошибке при попытке запроса истории бронирования с использованием утилит и моделей базы данных.

3.2.12 Тестирование интеграции select_date.py с select_tables.py с reservation_routes.py

Тест I31: Тест Интеграции Выбора Даты и Стола с Модулем Бронирования

- **Тип теста:** Позитивный
- **Описание:** Проверка передачи корректных данных о выбранной дате и столе в процессе бронирования.
- **Входные данные:**
  - Выбранная дата: `selected_date = "2024-01-01"`.
  - Выбранный стол: `selected_table = 1`.
- **Шаги:**
  1. Вызов функции выбора стола: `select_tables.select_tables(selected_date, selected_table)`.
  2. Проверка корректной передачи данных в модуль бронирования.
- **Результат:** Система должна корректно передать выбранную дату и стол в процессе бронирования.

Тест I32: Тест Интеграции Выбора Даты и Стола с Модулем Бронирования

- **Тип теста:** Негативный
- **Описание:** Проверка корректной обработки ошибок при передаче некорректных данных о выбранной дате и столе в процессе бронирования.
- **Входные данные:**

- Некорректная дата: `selected_date = "invalid_date"`.
- Некорректный стол: `selected_table = 0`.

- **Шаги:**

1. Попытка вызова функции выбора стола с некорректными данными: `select_tables.se selected_table)`.
2. Проверка сообщения об ошибке.

- **Результат:** Система должна корректно обработать ошибки и вывести сообщения об ошибке при передаче некорректных данных о выбранной дате и столе в процессе бронирования.

3.3 Аттестационное тестирование

3.3.1 Тестирование регистрации и входа в систему

Тест А1: Тест регистрации пользователя

- **Тип теста:** Позитивный
- **Описание:** Проверка успешной регистрации пользователя с корректными данными.
- **Входные данные:**
  - Корректное имя пользователя: `username = "valid_username"`.
  - Корректный пароль: `password = "valid_password"`.
  - Допустимый адрес электронной почты: `email = "valid@example.com"`.
- **Шаги:**

1. Попытка регистрации с корректными данными: `auth_routes.register(username, password, email)`.
2. Проверка успешной регистрации.

- **Результат:** Система должна успешно зарегистрировать пользователя с корректными данными.

Тест А2: Тест регистрации пользователя

- **Тип теста:** Негативный
- **Описание:** Проверка отображения сообщения об ошибке при попытке регистрации с недопустимыми данными.
- **Входные данные:**
  - Некорректное имя пользователя: `username = .`
  - Некорректный пароль: `password = .`
  - Недопустимый адрес электронной почты: `email = "invalid_email"`.
- **Шаги:**
  1. Попытка регистрации с некорректными данными: `auth_routes.register(username, password, email)`.
  2. Проверка сообщения об ошибке.
- **Результат:** Система должна отобразить сообщение об ошибке при попытке регистрации с недопустимыми данными.

Тест А3: Тест входа в систему

- **Тип теста:** Позитивный
- **Описание:** Проверка успешного входа пользователя с корректными учетными данными.
- **Входные данные:**
  - Зарегистрированное имя пользователя: `registered_username = "registered_user"`.
  - Пароль для зарегистрированного пользователя: `registered_password = "registered`
- **Шаги:**
  1. Попытка входа с корректными учетными данными: `auth_routes.login(registered_u registered_password)`.
  2. Проверка успешного входа.
- **Результат:** Система должна успешно войти в систему с корректными учетными данными.

Тест А4: Тест входа в систему

- **Тип теста:** Негативный
- **Описание:** Проверка отображения сообщения об ошибке при входе с недопустимыми данными.
- **Входные данные:**
  - Некорректное имя пользователя: `invalid_username = "nonexistent_user"`.
  - Некорректный пароль: `invalid_password = "wrong_password"`.
- **Шаги:**
  1. Попытка входа с недопустимыми учетными данными: `auth_routes.login(invalid_us invalid_password)`.
  2. Проверка сообщения об ошибке.
- **Результат:** Система должна отобразить сообщение об ошибке при входе с недопустимыми данными.

3.3.2 Тестирование просмотра дашборда

Тест А5: Тест Просмотра Дашборда пользователя

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного отображения дашборда пользователя после входа в систему.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными.
- **Шаги:**
  1. Вход пользователя в систему: `auth_routes.login(registered_username, registere`
  2. Переход на страницу дашборда.
  3. Проверка корректного отображения дашборда.
- **Результат:** Система должна корректно отобразить дашборд пользователя после входа в систему.

Тест А6: Тест наличия элементов на дашборде

- **Тип теста:** Позитивный
- **Описание:** Проверка наличия необходимых элементов на дашборде пользователя.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными.
- **Шаги:**
  1. Вход пользователя в систему: `auth_routes.login(registered_username, registere`
  2. Переход на страницу дашборда.
  3. Проверка наличия кнопок для бронирования, просмотра истории и других необходимых элементов.
- **Результат:** Дашборд пользователя должен содержать все необходимые элементы.

Тест А7: Тест Просмотра дашборда (Неавторизованный пользователь)

- **Тип теста:** Негативный
- **Описание:** Проверка отображения сообщения об ошибке при попытке доступа к дашборду без авторизации.
- **Входные данные:** Неавторизованный пользователь.
- **Шаги:**
  1. Переход на страницу дашборда без входа в систему.
  2. Проверка отображения сообщения об ошибке.
- **Результат:** Система должна отобразить сообщение об ошибке при попытке доступа к дашборду без авторизации.

3.3.3 Тестирование бронирования стола

Тест А8: Тест полного цикла бронирования

- **Тип теста:** Позитивный

- **Описание:** Проверка полного цикла бронирования от выбора даты до подтверждения.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными.
- **Шаги:**
  1. Вход пользователя в систему: `auth_routes.login(registered_username, registere`
  2. Выбор даты для бронирования: `reservation_routes.select_date(selected_date)`.
  3. Выбор стола для бронирования: `reservation_routes.select_tables(selected_tabl`
  4. Подтверждение бронирования: `reservation_routes.confirm_reservation()`.
  5. Проверка успешного бронирования в истории пользователя.
- **Результат:** Система должна успешно обработать все этапы бронирования, и пользователь должен видеть успешное бронирование в своей истории.

Тест А9: Тест бронирования (Все столы заняты)

- **Тип теста:** Негативный
- **Описание:** Проверка отображения ошибки при попытке бронирования, если все столы заняты.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными.
- **Шаги:**
  1. Занять все столы предварительно.
  2. Попытка бронирования стола: `reservation_routes.select_tables(selected_table)`
  3. Проверка отображения сообщения об ошибке.
- **Результат:** Система должна отобразить сообщение об ошибке при попытке бронирования, если все столы заняты.

3.3.4 Тестирование подтверждения бронирования

Тест A10: Тест подтверждения бронирования

- **Тип теста:** Позитивный
- **Описание:** Тестирование подтверждения бронирования пользователем.
- **Входные данные:** Зарегистрированный пользователь подтверждает бронирование.
- **Шаги:**
  1. Подтверждение бронирования: `reservation_routes.confirm_reservation()`.
  2. Проверка отображения подтвержденного бронирования в истории пользователя.
- **Результат:** Система должна успешно подтвердить бронирование пользователем, и бронирование должно отображаться в истории пользователя.

Тест A11: Тест подтверждения бронирования (стол занят)

- **Тип теста:** Негативный
- **Описание:** Проверка отображения сообщения об ошибке при попытке подтвердить бронирование, которое уже было подтверждено.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными и подтвержденным бронированием.
- **Шаги:**
  1. Попытка подтверждения бронирования: `reservation_routes.confirm_reservation(`
  2. Проверка отображения сообщения об ошибке.
- **Результат:** Система должна отобразить сообщение об ошибке при попытке подтвердить бронирование, которое уже было подтверждено.

Тест A12: Тест уведомления о подтвержденном бронировании

- **Тип теста:** Позитивный
- **Описание:** Проверка уведомления пользователя о подтвержденном бронировании.

- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными и подтвержденным бронированием.
- **Шаги:**
  1. Подтверждение бронирования: `reservation_routes.confirm_reservation()`.
  2. Проверка получения уведомления о подтвержденном бронировании.
- **Результат:** Пользователь должен успешно получить уведомление о подтвержденном бронировании.

Тест A13: Тест отображения подтвержденных бронирований в истории пользователя

- **Тип теста:** Позитивный
- **Описание:** Тестирование отображения подтвержденных бронирований в истории пользователя.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными и подтвержденным бронированием.
- **Шаги:**
  1. Вход пользователя в систему: `auth_routes.login(registered_username, registere`
  2. Подтверждение бронирования: `reservation_routes.confirm_reservation()`.
  3. Проверка отображения подтвержденного бронирования в истории пользователя.
- **Результат:** Подтвержденное бронирование должно отображаться в истории пользователя.

3.3.5 Тестирование панели администратора

Тест A14: Тест Удаления Бронирования из Панели Администратора

- **Тип теста:** Позитивный
- **Описание:** Проверка возможности администратора удалить бронирование из панели администратора.

- **Входные данные:** Зарегистрированный администратор с допустимыми учетными данными.
- **Шаги:**
    1. Вход администратора в систему: `auth_routes.login(admin_username, admin_passwo`
    2. Удаление бронирования из панели администратора: `admin_routes.delete_reservati`
    3. Проверка отсутствия бронирования в системе.
- **Результат:** Бронирование должно быть успешно удалено из системы.

Тест A15: Тест удаления несуществующего бронирования из панели администратора

- **Тип теста:** Негативный
- **Описание:** Проверка отображения сообщения об ошибке при попытке администратора удалить несуществующее бронирование из панели администратора.
- **Входные данные:** Зарегистрированный администратор с допустимыми учетными данными.
- **Шаги:**
    1. Вход администратора в систему: `auth_routes.login(admin_username, admin_passwo`
    2. Попытка удаления несуществующего бронирования: `admin_routes.delete_nonexiste`
    3. Проверка отображения сообщения об ошибке.
- **Результат:** Система должна отобразить сообщение об ошибке при попытке удаления несуществующего бронирования.

Тест A16: Тест отображения истории бронирования в панели администратора

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного отображения истории бронирования пользователей в панели администратора.
- **Входные данные:** Зарегистрированный администратор с допустимыми учетными данными.

- **Шаги:**

  1. Вход администратора в систему: `auth_routes.login(admin_username, admin_passwo`
  2. Переход к истории бронирования: `admin_routes.view_reservation_history()`.
  3. Проверка корректного отображения истории бронирования.

- **Результат:** История бронирования пользователей должна быть корректно отображена в панели администратора.

3.3.6 Тестирование истории бронирования

Тест A17: Тест Отображения Истории Бронирования Пользователя

- **Тип теста:** Позитивный
- **Описание:** Проверка корректного отображения истории бронирования пользователя.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными и хотя бы одним предыдущим бронированием.
- **Шаги:**

  1. Вход пользователя в систему: `auth_routes.login(user_username, user_password)`.
  2. Переход к истории бронирования: `reservation_history_routes.view_reservation_h`
  3. Проверка корректного отображения истории бронирования.

- **Результат:** История бронирования пользователя должна быть корректно отображена.

3.3.7 Тестирование отправки уведомлений

Тест A18: Тест Отправки Уведомлений Пользователю

- **Тип теста:** Позитивный
- **Описание:** Проверка отправки уведомлений пользователю после успешного бронирования.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными.

- **Шаги:**
  1. Выполнение успешного бронирования: `reservation_routes.complete_reservation()`
  2. Проверка отправки уведомления пользователю.
- **Результат:** Пользователь должен получить уведомление о успешном бронировании.

Тест A19: Тест отправки уведомлений пользователю

- **Тип теста:** Негативный
- **Описание:** Проверка отсутствия отправки уведомлений, если бронирование не выполнено.
- **Входные данные:** Зарегистрированный пользователь с допустимыми учетными данными.
- **Шаги:**
  1. Попытка неудачного бронирования: `reservation_routes.fail_reservation()`.
  2. Проверка отсутствия уведомления пользователю.
- **Результат:** Пользователь не должен получить уведомления в случае неудачного бронирования.

3.4 Нагрузочное тестирование

3.4.1 Тест L1: Тест одновременного бронирования

- **Тип теста:** Общий
- **Описание:** Тестирование системы на одновременное бронирование столов разными пользователями.
- **Входные данные:** Система работает в нормальном режиме.
- **Шаги:**
  1. Запуск сценария бронирования столов разными пользователями одновременно.
  2. Замер времени отклика системы.

3. Анализ загрузки сервера и базы данных.

- **Результат:** Оценка производительности системы при одновременном бронировании.

3.4.2 Тест L2: Тест авторизации

- **Тип теста:** Статический
- **Описание:** Тестирование авторизации большого числа пользователей одновременно.
- **Входные данные:** Система работает в статической нагрузке с большим числом пользователей.
- **Шаги:**
  1. Запуск сценария авторизации множества пользователей.
  2. Замер времени отклика системы.
  3. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка стабильности системы при авторизации большого числа пользователей.

3.4.3 Тест L3: Тест истории бронирований

- **Тип теста:** Пиковый
- **Описание:** Тестирование работы системы при большом объеме данных в истории бронирований.
- **Входные данные:** Увелчение объема данных в истории бронирований до максимального.
- **Шаги:**
  1. Запуск сценария просмотра истории бронирований.
  2. Замер времени отклика системы.
  3. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка производительности системы при просмотре большого объема исторических данных.

3.4.4 Тест L4: Тест обработки множественных запросов на подтверждение

- **Тип теста:** Общий
- **Описание:** Тестирование обработки системой множественных запросов на подтверждение бронирования.
- **Входные данные:** Система работает в нормальном режиме.
- **Шаги:**
  1. Постепенное увеличение числа одновременных пользователей подтверждающих заявки.
  2. Замер времени отклика системы.
  3. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка производительности системы при обработке множественных запросов администраторов.

3.4.5 Тест L5: Тест регистрации и бронирования

- **Тип теста:** Статический
- **Описание:** Тестирование системы при постоянной нагрузке регистрации и бронирования.
- **Входные данные:** Система работает в статической нагрузке с постоянным потоком новых пользователей и бронирований.
- **Шаги:**
  1. Запуск сценария постоянной регистрации новых пользователей.
  2. Запуск сценария бронирования столов в непрерывном режиме.
  3. Замер времени отклика системы.
  4. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка пропускной способности системы при постоянной нагрузке регистрации и бронирования.

3.4.6 Тест L6: Тест подтверждения бронирования

- **Тип теста:** Пиковый
- **Описание:** Система работает в пиковой нагрузке с большим числом запросов на подтверждение.
- **Входные данные:** Запуск максимального числа запросов на подтверждение бронирования.
- **Шаги:**
  1. Запуск сценария отправки максимального числа запросов на подтверждение.
  2. Замер времени отклика системы.
  3. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка производительности системы при пиковой нагрузке запросов на подтверждение.

3.4.7 Тест L7: Тест поддержания долговременной нагрузки

- **Тип теста:** Общий
- **Описание:** Тестирование системы на стабильность при продолжительной нагрузке.
- **Входные данные:** Система работает в течение продолжительного времени с переменной нагрузкой.
- **Шаги:**
  1. Запуск сценария с переменной интенсивностью регистрации и бронирования.
  2. Замер времени отклика системы.
  3. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка стабильности системы при долговременной переменной нагрузке.

3.4.8 Тест L8: Тест обработки уведомлений

- **Тип теста:** Статический
- **Описание:** Тестирование системы при постоянной нагрузке отправки уведомлений.
- **Входные данные:** Система работает в статической нагрузке с постоянным потоком уведомлений.
- **Шаги:**
  1. Запуск сценария постоянной отправки уведомлений пользователям.
  2. Замер времени отклика системы.
  3. Анализ загрузки сервера и базы данных.
- **Результат:** Оценка стабильности системы при постоянной нагрузке отправки уведомлений.

3.5 Покрытие кода тестами

$$Tcov = \left(\frac{Ltc}{Lcode}\right) \times 100\%$$

где

Tcov - тестовое покрытие

Ltc - количество строк кода покрытое тестами

Lcode - Общее количество строк кода

Тогда: **Tcov = (432/902) * 100% = 47,89%**

Глава 4

Журнал тестирования

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B1	Успешный вход в систему	Позитивный	
B2	Сообщение об ошибке о некорректном имени пользователя	Негативный	
B3	Сообщение об ошибке о некорректном пароле	Негативный	
B4	Перенаправление на целевую страницу	Позитивный	
B5	Успешная регистрация в системе	Позитивный	
B6	Сообщение об ошибке о существующем имени пользователя	Негативный	
B7	Сообщения об ошибке о некорректных данных	Негативный	
B8	Перенаправление на целевую страницу	Позитивный	
B9	Корректный выход из системы и перенаправление на главную страницу	Позитивный	

Таблица 4.1: Блочные тесты

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B10	Корректное отображение страницы выбора даты	Позитивный	
B11	Корректное отображение страницы выбора столов	Позитивный	
B12	Успешное бронирование столов	Позитивный	
B13	Сообщение об ошибке о занятости стола	Негативный	№1 Повторная запись сохраняется, нет сообщения об ошибке
B14	Корректная обработка запроса и изменение статуса бронирования	Позитивный	
B15	Корректное отображение панели пользователя с информацией о бронированиях	Позитивный	

Таблица 4.2: Блочные тесты (продолжение)

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B16	Корректное отображение панели администратора с обзором пользователей и бронированиями	Позитивный	
B17	Отсутствие доступа к панели администратора для обычного пользователя	Позитивный	
B17	Корректная обработка запроса и удаление бронирования из системы	Позитивный	
B18	Отсутствие доступа и корректное сообщение об ошибке	Негативный	№2 Удаление происходит, нет сообщения об ошибке

Таблица 4.3: Блочные тесты (продолжение)

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B19	Корректное отображение страницы с историей бронирования пользователя	Позитивный	
B20	Отсутствие доступа и корректное сообщение об ошибке	Негативный	

Таблица 4.4: Блочные тесты (продолжение)

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B21	Сгенерированное письмо соответствует ожидаемому формату и содержанию	Позитивный	
B22	Генерация завершается ошибкой и возвращается соответствующее сообщение	Негативный	

Таблица 4.5: Блочные тесты (продолжение)

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B23	Записи пользователя и бронирования успешно добавлены в базу данных	Позитивный	
B24	Получены корректные данные о пользователе и забронированных столах	Позитивный	
B25	Создание пользователя завершается ошибкой и возвращается соответствующее сообщение	Негативный	
B26	Возвращается сообщение об ошибке отсутствия данных	Негативный	

Таблица 4.6: Блочные тесты (продолжение)

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
B27	Метод возвращает корректный результат - пользователь является администратором	Позитивный	
B28	Метод возвращает корректный результат - пользователь не является администратором	Позитивный	
B29	Метод возвращает корректный результат - у пользователя есть активные бронирования	Позитивный	
B30	Метод возвращает корректный результат - у пользователя нет активных бронирований	Позитивный	

Таблица 4.7: Блочные тесты (продолжение)

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
В31	Запись пользователя корректно создана и сохранена	Позитивный	
В32	Запись пользователя не сохранена из-за нарушения уникальности имени пользователя	Позитивный	
В33	Запись бронирования корректно создана и сохранена	Позитивный	
В34	Запись бронирования не сохранена из-за отсутствия обязательных полей	Позитивный	

Таблица 4.8: Тестирование моделей User и Reservation

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I1	Запись нового пользователя с именем "new_user"корректно создана и сохранена в базе данных	Позитивный	
I2	Аутентификация прошла успешно, и доступ к данным пользователя с именем "auth_user"в базе данных осуществлен	Позитивный	
I3	Система обнаруживает существующее имя пользователя "existing_user"и возвращает сообщение об ошибке	Негативный	

Таблица 4.9: Тестирование функциональности регистрации и аутентификации

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I4	Пользователь успешно выбирает дату и стол после аутентификации	Позитивный	
I5	Отображение корректных данных пользователя, включая последнее бронирование, на странице панели пользователя	Позитивный	
I6	Пользователь получает сообщение об ошибке, указывающее, что выбранный стол уже занят, и бронирование не выполняется	Негативный	

Таблица 4.10: Тестирование функциональности выбора даты и стола, а также отображения данных пользователя

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I7	Бронирование успешно добавлено в историю, и оно корректно отображается при просмотре истории бронирования	Позитивный	
I8	История бронирования успешно отображается, и пользователь видит свои предыдущие бронирования	Позитивный	
I9	Система отображает сообщение об ошибке, указывающее на то, что выбранный стол уже занят	Негативный	№3 Некорректные данные сохраняются
I10	Система должна перенаправить пользователя на страницу входа и отобразить сообщение о необходимости аутентификации	Негативный	

Таблица 4.11: Тестирование функциональности истории бронирования

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I11	Система корректно отобразила панель администратора с данными о пользователях	Позитивный	
I12	Система успешно удалила бронирование с указанным идентификатором	Позитивный	
I13	Система перенаправила пользователя на главную страницу с сообщением о нехватке прав доступа	Негативный	№4 Перенаправления не происходит
I14	Система отобразила сообщение об ошибке, указывающее на нехватку прав доступа для выполнения операции	Негативный	

Таблица 4.12: Тестирование функциональности администраторской панели

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I15	Система корректно отобразила информацию о бронированиях пользователей и позволила администратору управлять ими	Позитивный	
I16	Система отобразила сообщение об ошибке, указывающее на нехватку прав доступа для выполнения операции	Негативный	
I17	Система успешно отправила уведомление с подтверждением бронирования на адрес пользователя	Позитивный	
I18	Система обработала сценарий без бронирования и выдала сообщение об ошибке	Негативный	

Таблица 4.13: Тестирование функциональности отправки уведомлений о бронировании

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I19	Система корректно использовала утилиты и модели для регистрации, аутентификации и бронирования	Позитивный	
I20	Система корректно обработала ошибки утилит и моделей, выведя сообщения об ошибке	Негативный	

Таблица 4.14: Тестирование корректной работы утилит и моделей при бронировании

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I21	Пользователь успешно перешел от выбора даты к выбору стола	Позитивный	
I22	Система предотвратила переход от выбора даты к выбору стола и вывела сообщение об ошибке	Негативный	
I23	Система успешно сохранила выбранную дату и стол перед бронированием	Позитивный	
I24	Система корректно обработала ошибки и вывела сообщения об ошибке при некорректных данных	Негативный	

Таблица 4.15: Тестирование перехода между этапами бронирования и сохранения выбранных данных

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I25	Система успешно позволяет администратору просматривать и удалять запросы на удаление бронирования	Позитивный	
I26	Система корректно обрабатывает ошибки и выводит сообщения об ошибке при попытке просмотра и удаления запросов с некорректными данными	Негативный	
I27	Система успешно отправляет подтверждение бронирования пользователю после успешного выбора стола	Позитивный	
I28	Система корректно обрабатывает ошибки и выводит сообщения об ошибке при попытке отправки подтверждения бронирования с некорректными данными	Негативный	

Таблица 4.16: Тестирование отправки подтверждения бронирования

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I27	Система успешно отправляет подтверждение бронирования пользователю после успешного выбора стола	Позитивный	
I28	Система корректно обрабатывает ошибки и выводит сообщения об ошибке при попытке отправки подтверждения бронирования с некорректными данными	Негативный	
I29	Система успешно отображает историю бронирования с использованием утилит и моделей базы данных	Позитивный	
I30	Система корректно обрабатывает ошибки и выводит сообщения об ошибке при попытке запроса истории бронирования с использованием утилит и моделей базы данных	Негативный	

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
I31	Система успешно передает выбранную дату и стол в процессе бронирования	Позитивный	
I32	Система корректно обрабатывает ошибки и выводит сообщения об ошибке при передаче некорректных данных о выбранной дате и столе в процессе бронирования	Негативный	

Таблица 4.18: Тестирование интеграции выбора даты и стола с модулем бронирования

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A1	Система успешно зарегистрировала пользователя с корректными данными	Позитивный	
A2	Система отобразила сообщение об ошибке при попытке регистрации с недопустимыми данными	Негативный	
A3	Система успешно вошла в систему с корректными учетными данными	Позитивный	
A4	Система отобразила сообщение об ошибке при входе с недопустимыми данными	Негативный	

Таблица 4.19: Тестирование регистрации и входа пользователя

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A5	Система корректно отобразила дашборд пользователя после входа в систему	Позитивный	
A6	Дашборд пользователя содержит все необходимые элементы	Позитивный	
A7	Система отобразила сообщение об ошибке при попытке доступа к дашборду без авторизации	Негативный	

Таблица 4.20: Тестирование дашборда пользователя

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A8	Система успешно обработала полный цикл бронирования	Позитивный	
A9	Система отобразила сообщение об ошибке при попытке бронирования, если все столы заняты	Негативный	

Таблица 4.21: Тестирование полного цикла бронирования

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A10	Система успешно подтвердила бронирование пользователем, и бронирование отображается в истории	Позитивный	
A11	Система отобразила сообщение об ошибке при попытке подтвердить уже подтвержденное бронирование	Негативный	
A12	Пользователь успешно получил уведомление о подтвержденном бронировании	Позитивный	
A13	Подтвержденное бронирование успешно отображается в истории пользователя	Позитивный	

Таблица 4.22: Тестирование подтверждения бронирования и уведомлений

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
A14	Бронирование успешно удалено из системы	Позитивный	
A15	Система отобразила сообщение об ошибке при попытке удалить несуществующее бронирование	Негативный	
A16	История бронирования пользователей корректно отображена в панели администратора	Позитивный	
A17	История бронирования пользователя корректно отображена	Позитивный	
A18	Уведомление успешно отправлено пользователю	Позитивный	
A19	Уведомление не отправлено в случае неудачного бронирования	Позитивный	

Таблица 4.23: Тестирование отображения истории бронирования и отправки уведомлений пользователю

Номер теста	**Фактический результат**	**Результат теста**	**Ошибка**
L1	Система обработала одновременное бронирование	Общий	
L2	Авторизация прошла стабильно при большом числе пользователей	Статический	
L3	Система справилась с просмотром большого объема данных в истории	Пиковый	
L4	Обработка множественных запросов на подтверждение прошла успешно	Общий	
L5	Система справилась с постоянной нагрузкой регистрации и бронирования	Статический	
L6	Система успешно обработала пиковую нагрузку запросов на подтверждение	Пиковый	
L7	Система стабильно работала при долговременной переменной нагрузке	Общий	
L8	Система стабильно обрабатывала постоянную нагрузку отправки уведомле-	Статический	№5 Задержка отправки

Глава 5

Примеры тестов

5.1 Тест B8

```python
class TestAuthRoutes(TestCase):
    def create_app(self):
        app.config['TESTING'] = True
        app.config['WTF_CSRF_ENABLED'] = False
        app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:84569053@localhost/test_reserv'
        return app

    def setUp(self):
        db.create_all()

    def tearDown(self):
        db.session.remove()
        db.drop_all()

    def test_registration_redirect(self):
        # Попытка регистрации с корректными данными
        with self.client:
            response = self.client.post('/auth/register', data=dict(
                username='newusername',
                password='newpassword',
                confirm_password='newpassword'
            ), follow_redirects=True)

            # Проверка, что код ответа - успешный (200)
            assert response.status_code == 200

            # Проверка перенаправления на целевую страницу (например, дашборд)
            assert request.path == '/dashboard'
```

Рис. 5.1: Тест для I4.1

5.2 Тест B14

```python
class ConfirmationProcessTest(TestCase):
    def create_app(self):
        app.config['TESTING'] = True
        app.config['WTF_CSRF_ENABLED'] = False
        app.config['SQLALCHEMY_DATABASE_URI'] = 'postgresql://postgres:84569053@localhost/test_reserv'
        return app

    def setUp(self):
        db.create_all()

    def tearDown(self):
        db.session.remove()
        db.drop_all()

    def test_process_tables(self):
        # Подготовка данных для теста
        with self.app.app_context():
            connection = connect_db()

        # Авторизация пользователя
        with self.client:
            response = self.client.post('/auth', data=dict(
                username='sglypa',
                password='123'
            ), follow_redirects=True)

        # Отправка запроса на подтверждение бронирования
        with self.client:
            response = self.client.post('/process_tables', data=dict(
                selected_tables='1, 2, 3',
                selected_date='2023-01-01'
            ), follow_redirects=True)

            # Проверка, что код ответа - успешный (200)
            self.assertEqual(response.status_code, 200)
```

Рис. 5.2: Тест для I4.1

5.3 Тест I4

```
public class ReservationRoutesIntegrationTest {

    @Autowired
    private DataSource dataSource;

    @Before
    public void setUp() throws Exception {
        IDatabaseConnection dbConnection = new DatabaseConnection(dataSource.getConnection());
        dbConnection.getConfig().setProperty(DatabaseConfig.PROPERTY_DATATYPE_FACTORY, new PostgresqlDataTypeFactory());

        // DbUnit для загрузки начального состояния базы данных
        IDataSet dataSet = new FlatXmlDataSetBuilder().build(getClass().getResourceAsStream("/datasets/initial_dataset.xml"));
        DatabaseOperation.CLEAN_INSERT.execute(dbConnection, dataSet);
    }

    @Test
    @Sql("/scripts/test_data.sql")  // Загружаем дополнительные SQL-скрипты
    public void testSelectDateAndTablesAfterAuthentication() {
        String username = "sqlypa";
        String password = "123";
        String loginUrl = "http://localhost:" + port + "/login";
        ResponseEntity<String> loginResponse = restTemplate.postForEntity(loginUrl, new AuthRequest(username, password), String.class);

        // Проверка успешной аутентификации
        assertThat(loginResponse.getStatusCode().is2xxSuccessful()).isTrue();

        // Выбор даты (замените на реальный URL и данные)
        String selectDateUrl = "http://localhost:" + port + "/select_date";
        ResponseEntity<String> selectDateResponse = restTemplate.getForEntity(selectDateUrl, String.class);

        // Проверка успешного выбора даты
        assertThat(selectDateResponse.getStatusCode().is2xxSuccessful()).isTrue();
    }
}
```

Рис. 5.3: Тест для I4.1

5.4 Тест I11

```java
public class AdminRoutesIntegrationTest {

    @LocalServerPort
    private int port;

    @Autowired
    private TestRestTemplate restTemplate;

    @Test
    @Sql("/scripts/test_data.sql")
    public void testAdminAccessToUserData() {
        // Аутентификация администратора (замените на реальные учетные данные)
        String adminUsername = "admin";
        String adminPassword = "123";
        String loginUrl = "http://localhost:" + port + "/login";
        ResponseEntity<String> loginResponse = restTemplate.postForEntity(loginUrl, new AuthRequest(adminUsername, adminPassword), String.class);

        // Проверка успешной аутентификации администратора
        assertThat(loginResponse.getStatusCode().is2xxSuccessful()).isTrue();

        // Просмотр данных пользователей (замените на реальный URL)
        String adminDashboardUrl = "http://localhost:" + port + "/admin/dashboard";
        ResponseEntity<String> adminDashboardResponse = restTemplate.getForEntity(adminDashboardUrl, String.class);

        // Проверка успешного доступа администратора к данным пользователей
        assertThat(adminDashboardResponse.getStatusCode().is2xxSuccessful()).isTrue();
    }

    // Класс для передачи данных аутентификации в POST-запросе
    private static class AuthRequest {
        private String username;
        private String password;

        public AuthRequest(String username, String password) {
            this.username = username;
            this.password = password;
        }
    }
}
```

Рис. 5.4: Тест для I4.1

5.5 Тест L4

```python
def make_reservation(user_id, selected_date, selected_tables):
    url = "http://localhost:50000/process_tables"
    headers = {'Content-Type': 'application/x-www-form-urlencoded'}
    data = {'selected_date': selected_date, 'selected_tables': selected_tables}
    session = requests.Session()
    response = session.post(url, data=data, headers=headers)

    # Можете добавить дополнительные проверки или логирование на основе ответа, если нужно
    assert response.status_code == 200
    print(f"Пользователь {user_id} сделал бронирование")


# Количество одновременных пользователей
num_users = 100

# Список для хранения потоков
threads = []

selected_date = "2023-01-01"
selected_tables = "1,2,3"

# Время начала для измерения длительности теста
start_time = time.time()

# Создание потоков и начало бронирования
for user_id in range(1, num_users + 1):
    thread = threading.Thread(target=make_reservation, args=(user_id, selected_date, selected_tables))
    thread.start()
    threads.append(thread)

# Ожидание завершения всех потоков
for thread in threads:
    thread.join()

# Рассчет и вывод длительности теста
test_duration = time.time() - start_time
print(f"Тест завершен за {test_duration:.2f} секунд")
```

Рис. 5.5: Тест для I4.1

Глава 6

Журнал найденных ошибок

6.1 Ошибка №1

6.1.1 Тест B13: Попытка бронирования уже занятого стола и проверка сообщения об ошибке

Входные данные

Пользователь, авторизованный в системе, занятый стол.

Шаги

1. Попытка бронирования занятого стола.

Результат

Система не корректно обрабатывает попытку бронирования уже занятого стола. Ожидается сообщение об ошибке, однако оно не отображается.

Состояние

Исправлена

6.2 Ошибка №2

6.2.1 Тест B18: Попытка удаления бронирования обычным пользователем

Входные данные

Пользователь, авторизованный в системе, с запросом на удаление бронирования.

Шаги

1. Отправка запроса на удаление бронирования.

Результат

Система не предоставляет корректное сообщение об ошибке при попытке обычного пользователя удалить бронирование. Ожидается сообщение о нехватке прав доступа, однако такое сообщение не отображается.

Состояние

Исправлена

6.3 Ошибка №3

6.3.1 Тест I9: Тест добавления бронирования в историю (Неверные данные)

Входные данные

- username = "auth_user"
- password = "secure_password"
- selected_date = "2023-01-01"
- selected_table_unavailable = 1 (стол, который уже занят)

Шаги

1. Аутентификация пользователя: login(username, password).
2. Выбор даты и недоступного стола: select_date, selected_table_unavailable.
3. Попытка подтверждения бронирования: process_tables(selected_date, selected_table_unava

Результат

Система не корректно обрабатывает попытку добавления бронирования с неверными данными. Ожидается сообщение об ошибке, указывающее на то, что выбранный стол уже занят. Однако, система не отображает это сообщение.

Состояние

Исправлена

6.4 Ошибка №4

6.4.1 Тест I13: Тест доступа неадминистратора к данным пользователей

Входные данные

- regular_user_username = "regular_user"
- regular_user_password = "regular_password"

Шаги

1. Аутентификация обычного пользователя: login(regular_user_username, regular_user_passw
2. Попытка просмотра данных пользователей: admin_dashboard().

Результат

Система не корректно обрабатывает попытку доступа неадминистратора к данным пользователей. Ожидается перенаправление пользователя на главную страницу с сообщением о нехватке прав доступа, однако такое перенаправление не происходит.

Состояние

Исправлена

6.5 Ошибка №5

6.5.1 Тест L8: Тест обработки уведомлений

Тип теста

Статический

Описание

Тестирование системы при постоянной нагрузке отправки уведомлений.

Входные данные

Система работает в статической нагрузке с постоянным потоком уведомлений.

Шаги

1. Запуск сценария постоянной отправки уведомлений пользователям.
2. Замер времени отклика системы.
3. Анализ загрузки сервера и базы данных.

Результат

В ходе тестирования выявлена нестабильность системы при постоянной нагрузке отправки уведомлений. Произошли случаи задержек в обработке уведомлений, что может привести к потере или задержке в доставке сообщений пользователям.

Состояние

Не исправлена (недостаточная мощность оборудования/сервера)

Глава 7

Результаты

В процессе выполнения работы объект тестирования был полностью проанализирован, включая его структуру, модули, методы и функциональности.

Разработанная стратегия и план тестирования включали в себя блочное, интеграционное, аттестационное и нагрузочное тестирование. Критерии прохождения тестирования были определены, а также установлены условия возобновления и приостановки тестов. Было обеспечно систематическое покрытие всех функциональных аспектов приложения.

В ходе работы не была исправлена ошибка вызывающая нестабильную работу подсистемы уведомлений, связанная с недостаточной мощностью оборудования/сервера. Журнал тестирования включает в себя результаты всех проведенных тестов, покрытие кода тестами, выявленные ошибки и их статус.

По итогу анализа результатов тестирования системы, ее можно назвать работоспособной, все заявленные функиональные возможности выполняются без ошибок.