МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Псковский государственный университет»

Передовая инженерная школа гибридных технологий

в станкостроении Союзного государства

Отделение информационных-коммуникационных технологий

Задание

по дисциплине «Методы и средства проектирования информационных систем и технологий»

Выполнил: Васильева К.А.

группа 0433-06

Проверил: Луканов С. Ю.

Псков

2024

Оглавление

[**Задание:** 2](#_Toc167828106)

[**Состав работы** 2](#_Toc167828107)

[**Idef0** 3](#_Toc167828108)

[**Idef3** 4](#_Toc167828109)

[**DFD** 4](#_Toc167828110)

[**Диаграммы UML (Use Cases Diagram + Class Diagram)** 5](#_Toc167828111)

[1) Use Cases Diagram 5](#_Toc167828112)

[2) Class Diagram 5](#_Toc167828113)

[**Схема БД** 8](#_Toc167828114)

[**Сквозные тесты:** 8](#_Toc167828115)

[Сценарий 1: регистрация нового тура для клиента AllTours. 8](#_Toc167828116)

[Сценарий 2: Регистрация нового клиента 9](#_Toc167828117)

[Сценарий 3: Оформление тура 10](#_Toc167828118)

[Сценарий 4: Поиск тура 11](#_Toc167828119)

[Сценарий 5: Оплата тура 11](#_Toc167828120)

[**Модульные тесты** 12](#_Toc167828121)

[1) tests/test\_views.py 12](#_Toc167828122)

[2) tests/test\_models.py 13](#_Toc167828123)

[3) tests/test\_forms.py 15](#_Toc167828124)

[4) tests/test\_database.py 16](#_Toc167828125)

[5) tests/test\_clients.py 17](#_Toc167828126)

[6) tests/test\_tours.py 18](#_Toc167828127)

[Итог: 19](#_Toc167828128)

[**Реализация взаимодействия с БД через SQL запросы, ORM** 19](#_Toc167828129)

# **Задание:**

Турфирма  
Разработать АИС для компании AllTours. Компания занимается составлением и организацией туров по различным направлениям. Туры разделяются на круизные, курортные, туристические, бизнес и эксклюзивные. Клиенты могут выбрать один из планов: AllTours Client, AllTours Business, AllTours World и AllTours Exclusive. Система должна автоматизировать процесс регистрации тура для клиента.  
БД должна содержать данные о клиентах, турах, договорах с перевозчиками и отелями, маршрутах, отзывах.

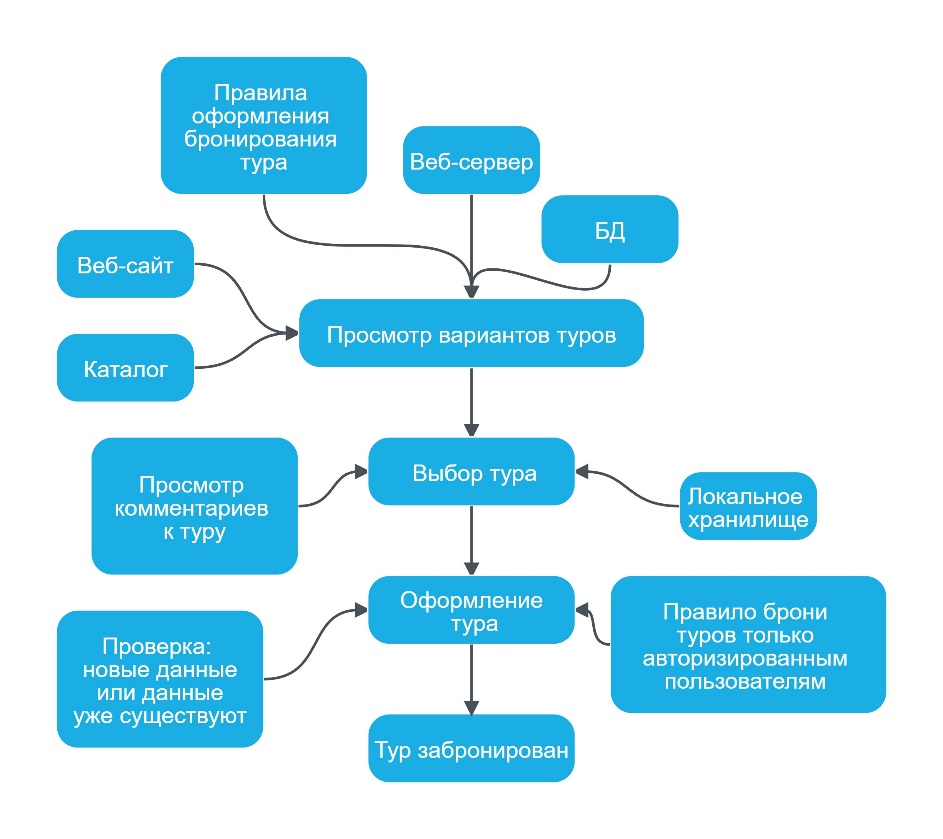
# **Состав работы**

1. Idef0, Idef3, DFD диаграммы
2. Диаграммы UML (Use Cases Diagram + Class Diagram)
3. Схема БД
4. GitHub репозиторий с двумя ветками
5. Сквозные и модульные тесты
6. Реализация взаимодействия с БД через SQL запросы
7. Реализация взаимодействия с БД через ORM
8. Реализация шаблона проектирования

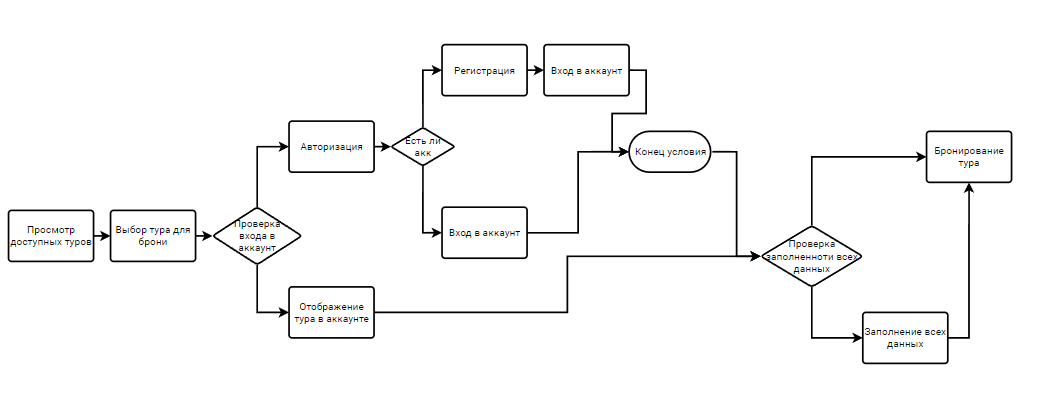
# **Idef0**



Детализация диаграммы:

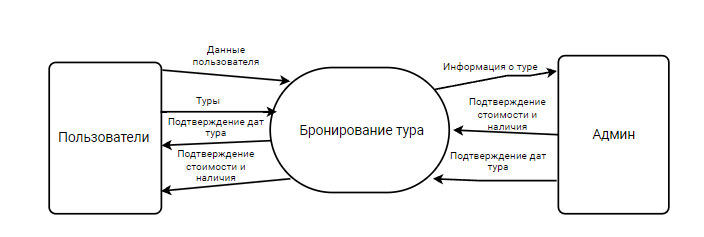


# **Idef3**



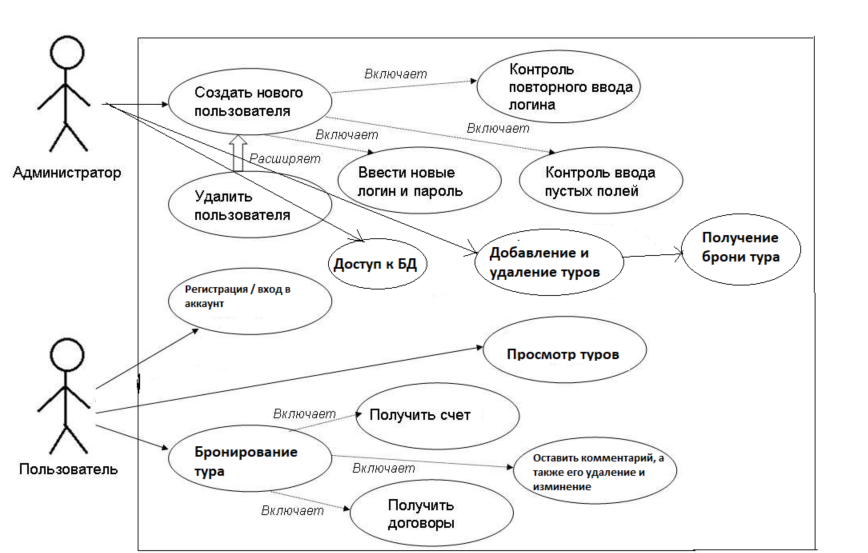
# **DFD**

Диаграмма процесса бронирования тура

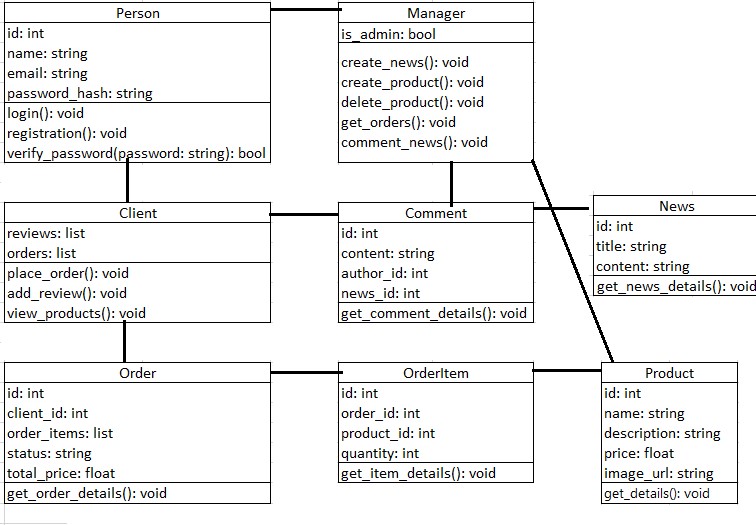


# **Диаграммы UML (Use Cases Diagram + Class Diagram)**

## 1) Use Cases Diagram



## 2) Class Diagram



Person (Человек)

Person

Атрибуты:

id: int

name: string

email: string

password\_hash: string

Методы:

login(): void

registration(): void

verify\_password(password: string): bool

Manager (Менеджер, наследуется от Person)

Атрибуты:

is\_admin: bool

Методы:

create\_news(): void

create\_product(): void

delete\_product(): void

get\_orders(): void

comment\_news(): void

Client (Клиент, наследуется от Person)

Client (наследуется от Person)

Атрибуты:

reviews: list

orders: list

Методы:

place\_order(): void

add\_review(): void

view\_products(): void

Product (Товар)

Product

Атрибуты:

id: int

name: string

description: string

price: float

image\_url: string

Методы:

get\_details(): void

Order (Заказ)

Order

Атрибуты:

id: int

client\_id: int

order\_items: list

status: string

total\_price: float

Методы:

get\_order\_details(): void

OrderItem (Товар в заказе)

OrderItem

Атрибуты:

id: int

order\_id: int

product\_id: int

quantity: int

Методы:

get\_item\_details(): void

Comment (Комментарий к новостям)

Comment

Атрибуты:

id: int

content: string

author\_id: int

news\_id: int

Методы:

get\_comment\_details(): void

News: Представляет новость, к которой могут быть оставлены комментарии.

comments (связь один ко многим с Comment)

News

News

Атрибуты:

id: int

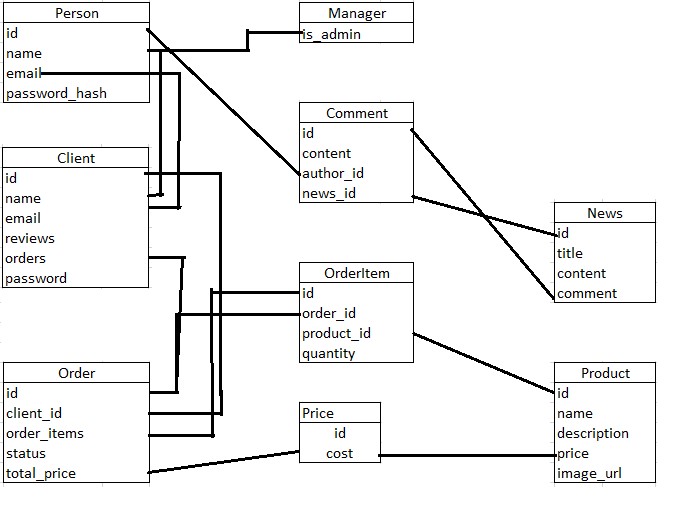
title: string

content: string

Методы:

get\_news\_details(): void

# **Схема БД**



**Сквозные тесты:**

Сценарий 1: регистрация нового тура для клиента AllTours.

1. Открытие страницы сайта «AllTours»

2. Переход на страницу регистрации тура для клиента из Header (при условии, что авторизация еще не пройдена)

3. Ввод данных для регистрации тура:

• Выбор типа тура (круизный, курортный, туристический, бизнес, эксклюзивный)

• Выбор плана (AllTours Client, AllTours Business, AllTours World, AllTours Exclusive)

4. Позитивный тест:

1. Выбрать тип тура

2. Выбрать план

3. Нажать на кнопку «Зарегистрировать тур»

4. Редирект на главную страницу

5. Проверить, что тур успешно зарегистрирован для выбранного клиента и отображается в его личном кабинете

5. Негативные тесты:

• Не выбран тип тура:

1. Не выбирать тип тура

2. Выбрать план

3. Нажать на кнопку «Зарегистрировать тур»

4. Проверить, что отображается сообщение об ошибке "Выберите тип тура"

• Не выбран план:

1. Выбрать тип тура

2. Не выбирать план

3. Нажать на кнопку «Зарегистрировать тур»

5. Проверить, что отображается сообщение об ошибке "Выберите план"

Такие сквозные тесты позволят проверить функционал регистрации новых пользователей и добавления товаров в корзину, а также зарегистрации новых туров для клиентов компании AllTours, включая различные сценарии с позитивными и негативными тестами.

## Сценарий 2: Регистрация нового клиента

1. Открытие главной страницы АИС «AllTours»

2. Переход на страницу регистрации из меню навигации

3. Ввод данных для регистрации:

• Поле ввода имени, фамилии, электронной почты, номера телефона, выбор плана (AllTours Client, AllTours Business, AllTours World, AllTours Exclusive)

• Позитивный тест:

1. Ввести имя, фамилию, корректный адрес электронной почты, номер телефона

2. Выбрать план AllTours Client

3. Нажать кнопку «Зарегистрироваться»

4. Проверить, что клиент успешно зарегистрирован и данные отображаются корректно

• Негативные тесты:

• Пустые поля:

1. Оставить пустым поле имени

2. Ввести фамилию, корректный адрес электронной почты, номер телефона

3. Выбрать план AllTours Business

4. Нажать кнопку «Зарегистрироваться»

5. Проверить, что отображается сообщение об ошибке "Заполните поле Имя"

• Некорректный адрес электронной почты:

1. Ввести имя, фамилию, некорректный адрес электронной почты

2. Ввести номер телефона

3. Выбрать план AllTours World

4. Нажать кнопку «Зарегистрироваться»

5. Проверить, что отображается сообщение об ошибке "Некорректный адрес электронной почты"

• Неправильный выбор плана:

1. Ввести имя, фамилию, корректный адрес электронной почты, номер телефона

2. Выбрать несуществующий план "AllTours Luxury"

3. Нажать кнопку «Зарегистрироваться»

4. Проверить, что отображается сообщение об ошибке "Выберите корректный план"

## Сценарий 3: Оформление тура

1. Авторизация в системе АИС «AllTours»

2. Выбор раздела "Оформление тура"

3. Ввод данных для оформления тура:

• Выбор направления тура, даты, продолжительности, выбор услуг (перевозка, проживание, экскурсии)

• Позитивный тест:

1. Выбрать круизное направление, дату, продолжительность 7 дней

2. Выбрать услуги: перевозку, проживание, экскурсии

3. Нажать кнопку «Оформить тур»

4. Проверить, что тур успешно оформлен и отображаются детали заказа

• Негативные тесты:

• Неправильная дата:

1. Выбрать курортное направление, некорректную дату

2. Выбрать продолжительность 10 дней

3. Выбрать услугу проживания

4. Нажать кнопку «Оформить тур»

5. Проверить, что отображается сообщение об ошибке "Выберите корректную дату"

• Отсутствие выбранных услуг:

1. Выбрать туристическое направление, дату, продолжительность 5 дней

2. Не выбирать ни одной услуги

3. Нажать кнопку «Оформить тур»

4. Проверить, что отображается сообщение об ошибке "Выберите хотя бы одну услугу"

Сценарий 4: Поиск тура

1. Авторизация в системе АИС «AllTours»

2. Переход на страницу поиска тура

3. Ввод критериев поиска:

• Выбор страны, города, даты, продолжительности, типа отдыха

• Позитивный тест:

1. Выбрать страну "Италия", город "Рим"

2. Указать дату начала тура и продолжительность 7 дней

3. Выбрать тип отдыха "Культурный"

4. Нажать кнопку "Поиск"

5. Проверить, что отображаются результаты поиска соответствующие критериям

• Негативные тесты:

• Пустой запрос:

1. Не выбирать никакие критерии поиска

2. Нажать кнопку "Поиск"

3. Проверить, что отображается сообщение об ошибке "Укажите хотя бы один критерий поиска"

• Несуществующие данные:

1. Ввести несуществующую страну или город

2. Указать некорректную дату или продолжительность

3. Нажать кнопку "Поиск"

4. Проверить, что отображается сообщение о том, что по вашему запросу ничего не найдено

Сценарий 5: Оплата тура

1. Авторизация в системе АИС «AllTours»

2. Выбор уже оформленного тура из личного кабинета

3. Переход на страницу оплаты

4. Ввод данных для оплаты:

• Выбор способа оплаты, ввод данных карты, подтверждение платежа

• Позитивный тест:

1. Выбрать способ оплаты через банковскую карту

2. Ввести номер карты, срок действия, CVV-код

3. Нажать кнопку "Оплатить"

4. Проверить, что платеж успешно проведен и отображается подтверждение

• Негативные тесты:

• Неверные данные карты:

1. Ввести неверный номер карты

2. Ввести некорректный срок действия или CVV-код

3. Нажать кнопку "Оплатить"

4. Проверить, что отображается сообщение об ошибке "Неверные данные карты"

• Отмена платежа:

1. Выбрать способ оплаты через интернет-банк

2. Начать процесс оплаты, но прервать его до подтверждения

3. Проверить, что отображается сообщение о прерывании процесса оплаты

**Модульные тесты**

Во всех тестах был использован метод SetUp

1. tests/test\_views.py

Файл test\_views.py содержит тесты для проверки представлений.

Количество тестов: 5

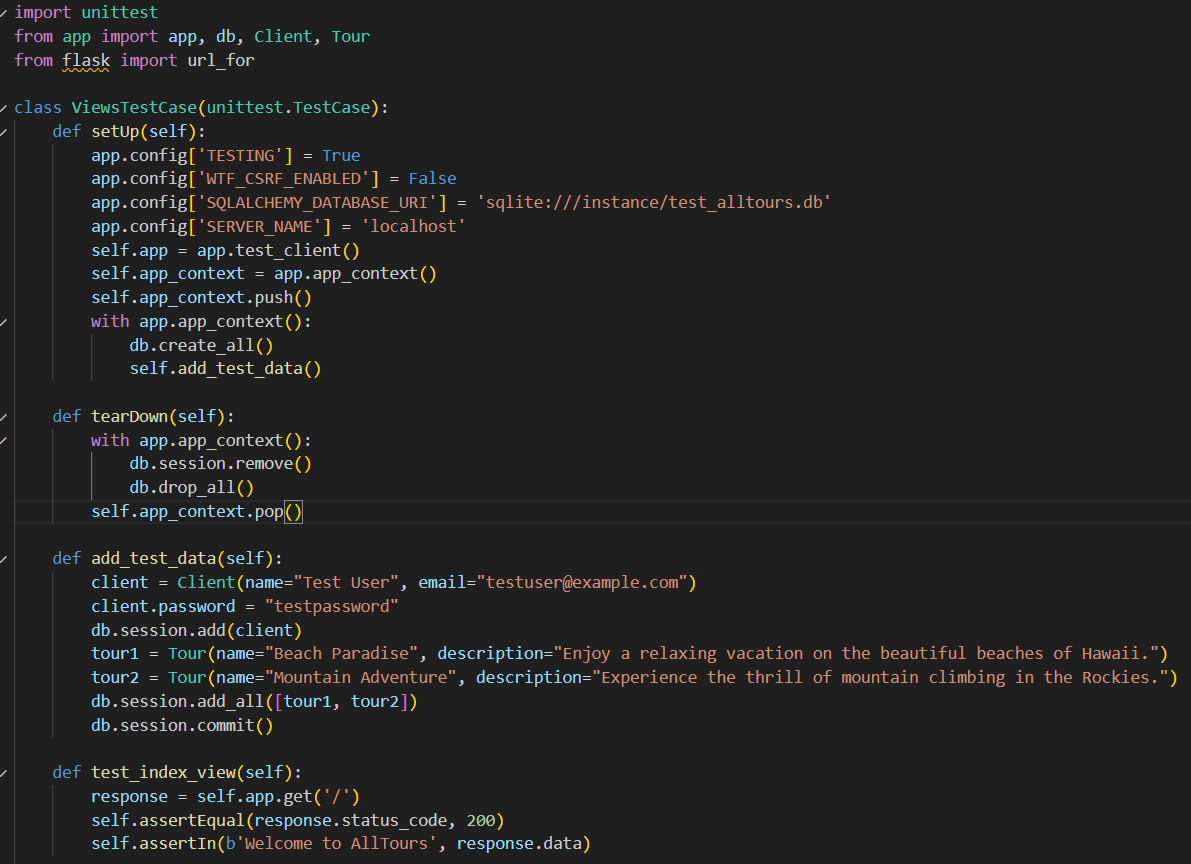
1 test\_index\_view: Проверяет, что главная страница загружается успешно и содержит текст "Welcome to AllTours".

2 test\_tour\_detail\_view: Проверяет, что страница деталей тура загружается успешно и содержит название тура.

3 test\_register\_view: Проверяет, что страница регистрации загружается успешно и содержит текст "Register".

4 test\_login\_view: Проверяет, что страница входа загружается успешно и содержит текст "Login".

5 test\_logout\_view: Проверяет, что после выхода сессия очищается, и пользователь перенаправляется на главную страницу.





1. tests/test\_models.py

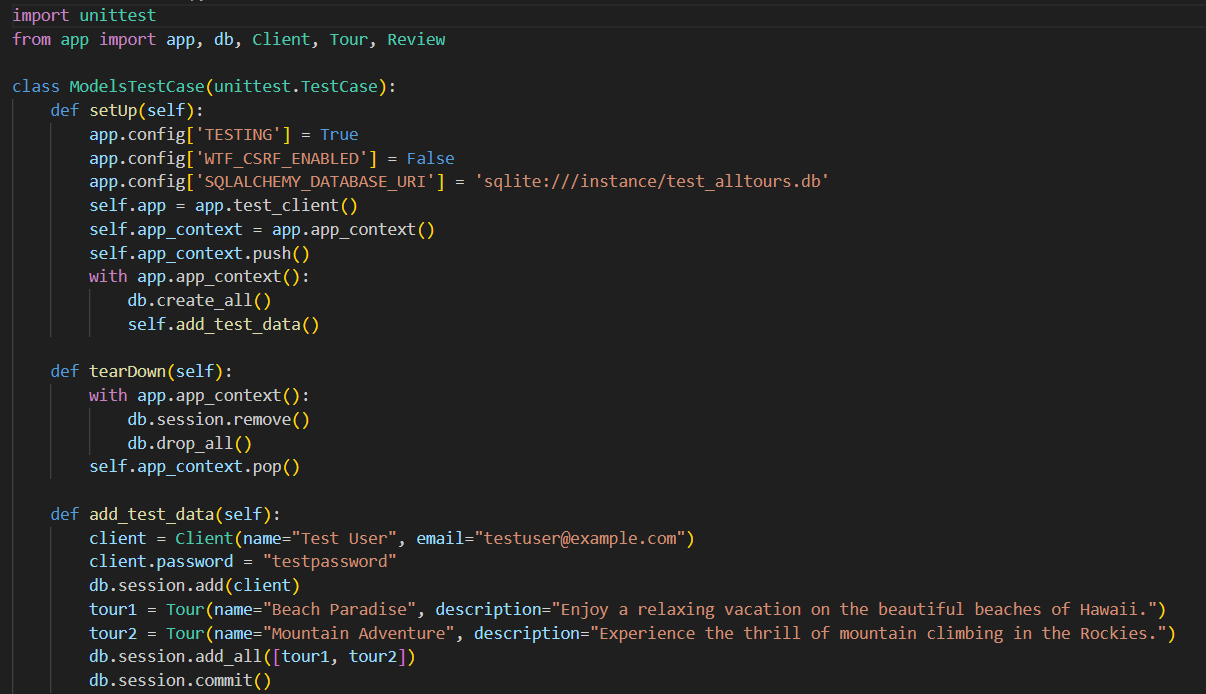
Файл test\_models.py содержит тесты для проверки моделей.

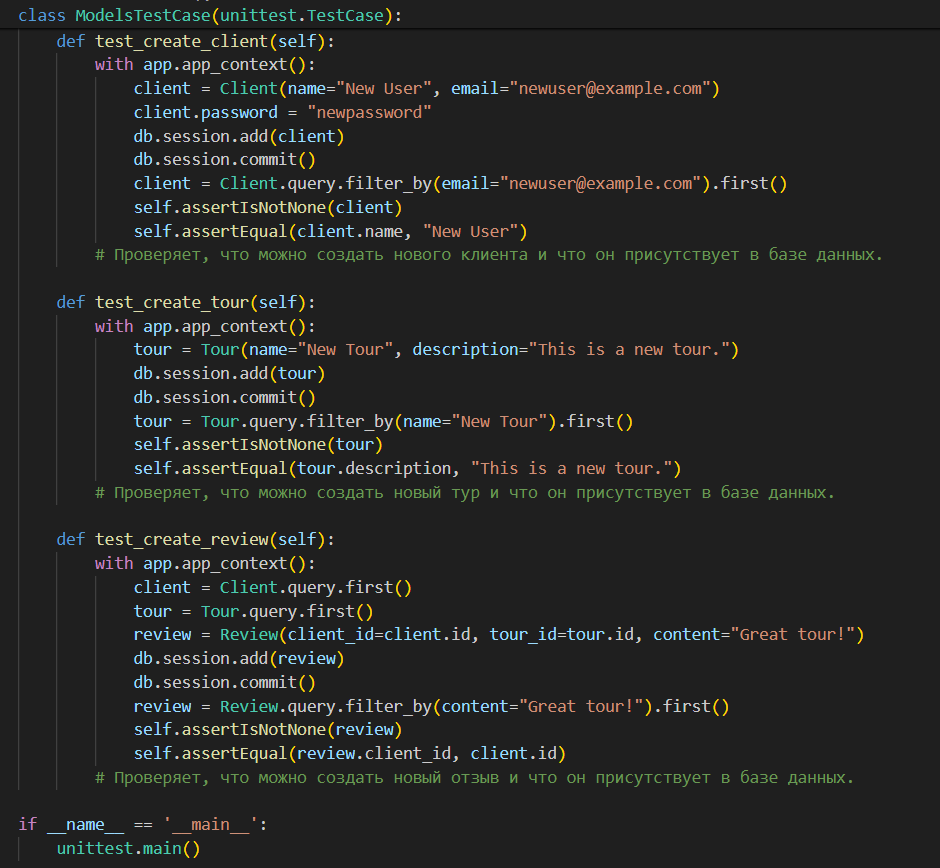
Количество тестов: 3

1 test\_create\_client: Проверяет, что можно создать нового клиента и что он присутствует в базе данных.

2 test\_create\_tour: Проверяет, что можно создать новый тур и что он присутствует в базе данных.

3 test\_create\_review: Проверяет, что можно создать новый отзыв и что он присутствует в базе данных.





1. tests/test\_forms.py

Файл test\_forms.py содержит тесты для проверки форм.

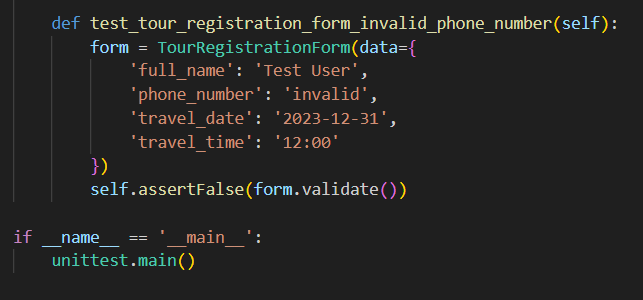
Количество тестов: 3

1 test\_tour\_registration\_form: Проверяет, что форма регистрации тура валидируется успешно с корректными данными.

2 test\_tour\_registration\_form\_missing\_full\_name: Проверяет, что форма регистрации тура не валидируется при отсутствии полного имени.

3 test\_tour\_registration\_form\_invalid\_phone\_number: Проверяет, что форма регистрации тура не валидируется при некорректном номере телефона.





1. tests/test\_database.py

Файл test\_database.py содержит тесты для проверки наличия базы данных, клиентов, туров и изображений.

Количество тестов: 4

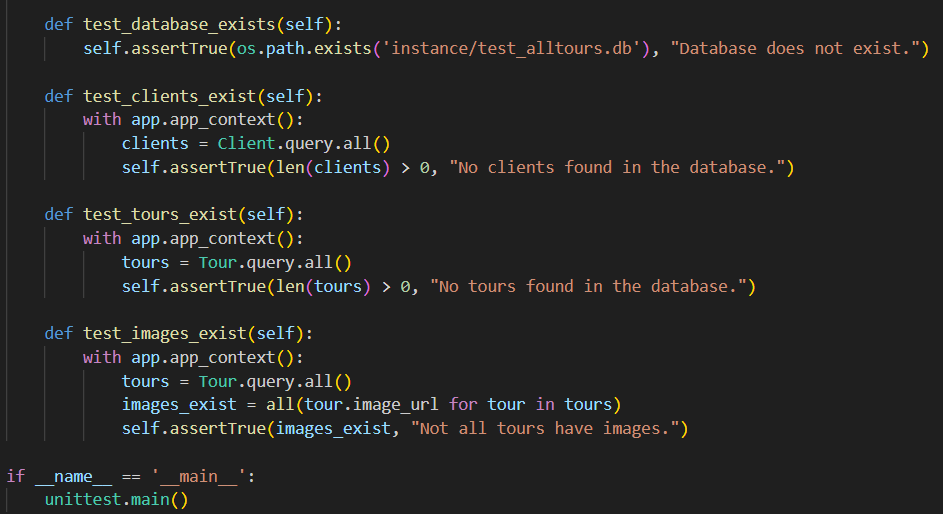
1 test\_database\_exists: Проверяет, существует ли база данных.

2 test\_clients\_exist: Проверяет, существуют ли клиенты в базе данных.

3 test\_tours\_exist: Проверяет, существуют ли туры в базе данных.

4 test\_images\_exist: Проверяет, имеют ли все туры изображения.



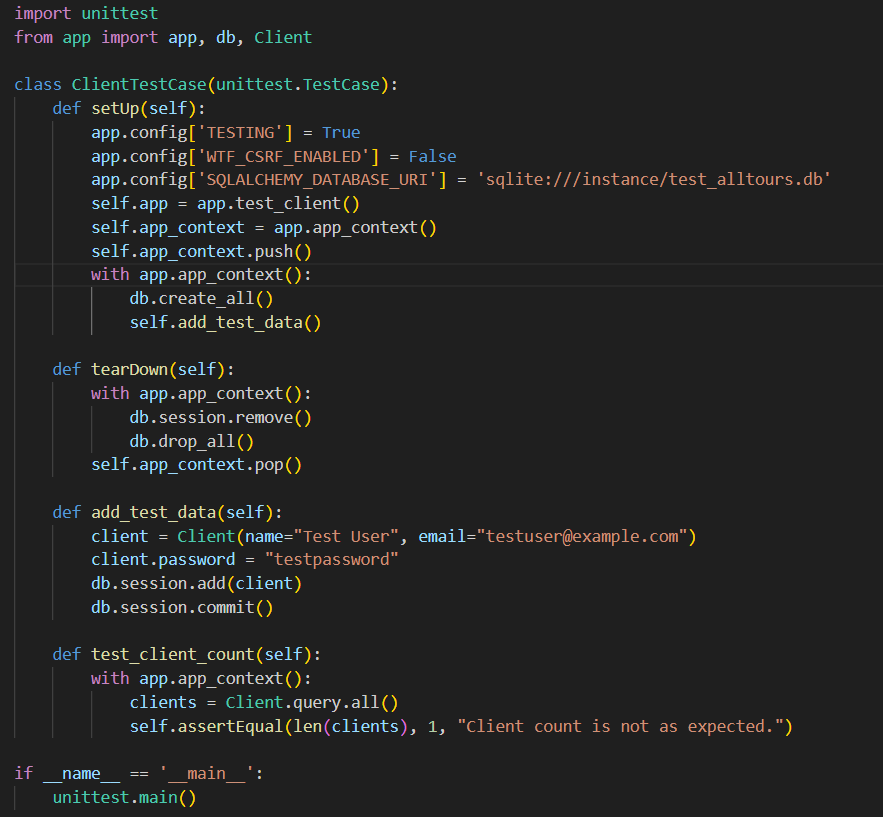


1. tests/test\_clients.py

Файл test\_clients.py содержит тесты для проверки клиентов в базе данных.

Количество тестов: 1

1 test\_client\_count: Проверяет, что количество клиентов в базе данных соответствует ожидаемому значению.



1. tests/test\_tours.py

Файл test\_tours.py содержит тесты для проверки туров в базе данных.

Количество тестов: 1

1 test\_tour\_count: Проверяет, что количество туров в базе данных соответствует ожидаемому значению.

tests/test\_images.py

Файл test\_images.py содержит тесты для проверки наличия изображений у туров.

Количество тестов: 1

1. test\_images\_exist: Проверяет, имеют ли все туры изображения.



Итог:

5 тестов в test\_views.py

3 теста в test\_models.py

3 теста в test\_forms.py

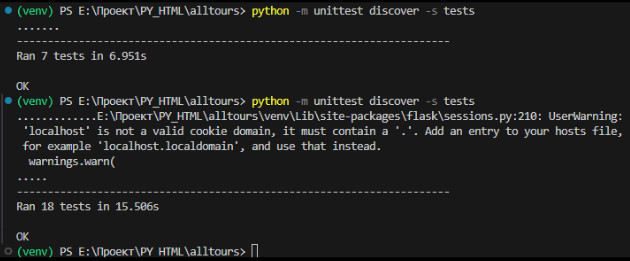
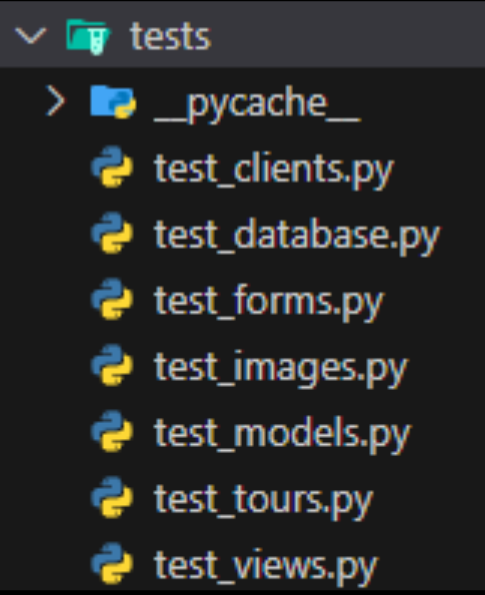
4 теста в test\_database.py

1 тест в test\_clients.py

1 тест в test\_tours.py

1 тест в test\_images.py

Общее количество тестов: 18



# **Реализация взаимодействия с БД через SQL запросы, ORM**

Код представляет модель данных для интернет-магазина туров. В нем определены следующие классы и их поля:

1. Person: Определяет общие поля для всех сущностей, связанных с конкретными людьми. Включает идентификатор, имя, email, и хэш пароля. Также содержит методы для работы с паролем.

2. Manager и Client: Наследуют от класса Person с добавлением дополнительных полей, таких как is\_admin для отметки администратора и связей с отзывами и заказами.

- lazy=True: Указывает, что связанные объекты будут загружаться лениво, то есть только при первом доступе к ним. Это помогает оптимизировать производительность, загружая данные только тогда, когда они действительно нужны.

3. Product: Представляет продукты в магазине. Включает поле для цены и изображения.

4. Order и OrderItem: Представляют заказы и элементы заказа соответственно. Связаны с клиентами и продуктами.

5. Comment и News: Комментарии и новости с возможностью комментирования аутентифицированными пользователями.

6. Tour и Review: Туры и отзывы о них. Отношение "многие к одному" между туром и клиентом через отзыв.

7. Registration и CartItem: Регистрация на туры и элементы корзины с полями для имени, номера телефона, даты и времени путешествия. Взаимосвязаны с соответствующими турами и клиентами.

Каждый класс представляет таблицу в базе данных, а поля классов являются столбцами в этих таблицах. Они описывают структуру данных и связи между ними для эффективного хранения информации о пользователях, продуктах, заказах, отзывах и других элементах системы.

Этот код является частью модели базы данных, используемой для работы с ORM (Object-Relational Mapping) в Flask-SQLAlchemy, что упрощает взаимодействие с базой данных.

