1. **Модульные тесты.** На уровне модульного тестирования основное внимание уделяется тестированию отдельных компонентов системы независимо друг от друга. Здесь мы сможем протестировать контроллер бэка, методы валидации данных и взаимодействие с репозиторием.
   1. **Тестирование валидации данных:**

* Если *card\_number* содержит строку или число, где больше 16 цифр, то тестируемый метод возвращает 400 с сообщением об ошибке;
* Если *card\_holder* содержит более одного пробела, то тестируемый метод возвращает 400 с сообщением об ошибке;
* Если *card\_expiration* содержит месяц и год меньшую текущего месяца и года или строку, то тестируемый метод возвращает 400 с сообщением об ошибке;
* Если *cvv* содержит строку или число более 3 цифр, то тестируемый метод возвращает 400 с сообщением об ошибке;
* Если *order\_number* содержит строку где больше 16 символов, то тестируемый метод возвращает 400 с сообщением об ошибке;
* Если *sum* содержит число или строка, которая не содержит запятых или число в котором после запятой больше двух символов, то тестируемый метод возвращает 400 с сообщением об ошибке;
  1. **Тестирование метода взаимодействия с репозиторием:**
* Проверка успешного сохранения данных, если все параметры правильные и метод **setOrderIsPaid** возвращает true.
* Тестирование исключений, выбрасываемых в случае несоответствия номера заказа и суммы (например, если сумма не совпадает с суммой, которую пришлёт сервис A).
* Тестирование обработки ошибок, если **setOrderIsPaid** возвращает false.
  1. **Тестирование взаимодействия с сервисом A**
* Заглушка, имитирующая успешный ответ сервиса A, который возвращает HTTP-код 200.
* Заглушка, имитирующая неудачный ответ сервиса A с HTTP-кодом 403.
  1. **Позитивный тест.** Цель теста: Убедиться, что метод обработки платежей правильно обрабатывает корректные данные и возвращает успешный ответ.

**Описание теста:** Имеется действующий аккаунт и достаточное количество средств. Сервис для обработки платежей доступен и функционирует корректно.

**Шаги:**

Определим объект данных для входа, содержащий корректную информацию о платеже. Например:

payment\_data = {

" *sum* ": 100.00,

" *order\_number* ": "444444444",

" *card\_holder* ": "Ivanov Ivan",

" *card\_number* ": "4111111111111111",

" *card\_expiration* ": "12/25",

" *cvv* ": "123"

}

Вызваем метод process\_payment(payment\_data).

Сохраним результат в переменной.

**Ожидаемый результат:**

Метод должен вернуть объект с полем status, равным "success", и полем confirmation\_id, содержащим уникальный идентификатор транзакции.

1. **Интеграционные тесты. Интеграционное тестирование направлено на проверку взаимодействия между различными модулями. В данном случае мы проверим, как бэк взаимодействует с фронтом, репозиторием и сервисом A.**

* Проверяем связку «бэк-фронт»:
  + Если *card\_holder* содержит более одного пробела, то, после получения ответа от бэка, на фронте выделяется поле «Номер карты»красной рамкой;
  + Если *card\_expiration* содержит месяц и год меньшую текущего месяца и года или строку, то, после получения ответа от бэка, на фронте выделяется поле «Год и месяц карты»красной рамкой;
  + Если *cvv* содержит строку или число более 3 цифр, то, после получения ответа от бэка, на фронте выделяется поле «CVV»красной рамкой;
  + Если *order\_number* содержит строку где больше 16 символов, то, после получения ответа от бэка, на фронте выделяется поле «Номер заказа»красной рамкой;
  + Если *sum* содержит число или строка, которая не содержит запятых или число в котором после запятой больше двух символов, то, после получения ответа от бэка, на фронте выделяется поле «Сумма»красной рамкой;
  + Отправка корректного JSON-объекта от фронта к бэку и проверка, что ответ бэка соответствует ожидаемому (код 200 и сообщение об успешной оплате).
  + Отправка некорректного объекта и проверка, что бэк возвращает ошибку с кодом 400 и соответствующим сообщением.
* Проверяем связку «бэк-api шлюз банка»:
  + Проверяем, доступен ли api шлюз. Если ответа от api нет, то, после получения ответа от бэка, на фронте выделяется сообщение «Банк не отвечает»;
  + Проверяем, ответ от api шлюза. Если код ответа 403, то, на фронте выделяется сообщение «Не достаточно средств»;
* Проверяем связку «бэк-репозиторий»:
  + Проверяем, ответ от api шлюза. Если код ответа от api шлюза банка 200 и если метод setOrderIsPaid возвращает false, то, после получения ответа от бэка, на фронте появляется alert окно с оповещением «Платеж прошел, но возникла ошибка базы сервиса, сообщите администратору». Выброшенное исключение записывается в error log.
  + Проверяем, ответ от api шлюза. Если код ответа от api шлюза банка 200 и если метод setOrderIsPaid возвращает true, то, после получения ответа от бэка, на фронте появляется alert окно с оповещением «Платеж успешно осуществлен»

1. **Системное тестирование.** Системное тестирование охватывает проверку всей системы в целом. Здесь мы должны убедиться, что все модули работают вместе согласованно и в соответствии с требованиями.
   1. **Тестирование полного рабочего потока оплаты:**
   * Отправка корректного запроса на оплату и проверка, что все компоненты системы работают правильно и оплата проходит успешно. Ожидаемый результат: успешный ответ от сервиса A, запись в БД и корректный ответ для фронта.
   1. **Тестирование обработки ошибок:**
   * Попытка сделать платёж с некорректными данными и проверка, что система правильно обрабатывает эту ситуацию, предоставляя пользователю понятное сообщение об ошибке.
   * Проверка, что при отказе сервиса A (HTTP-код 403) пользователю возвращается правильная информация и запись в БД не происходит
   1. **Тестирование нагрузки:**
   * Проверка системы под повышенной нагрузкой на несколько параллельных оплат, чтобы подтвердить, что система выдерживает нагрузку и работает эффективно