Содержание:

Задание 1:

Смотри отдельный jpg-файл ‘задание1’ в репозитории.

Описание задачи:

Сделайте модель в нотации BPMN 2.0 по описанию рабочего потока ниже.

Описание:
В компании «X» работает выдача ит-оборудования пользователям по заявке, которую надо оформлять в системе №1. Пользователь офиса, когда ему нужно, делает заявку через почту или по телефону в 1-ую линию технической поддержки на оборудование. Он получает номер обращения, благодаря которому может отследить, что происходит с его заявкой. Первая линия технической поддержки делает заявку на оборудование (на основании заявки от пользователя) в системе №2 отделу снабжения. Отдел снабжения принимает заявку в системе №2. Отдел снабжения видит, что есть заявка на оборудование по уведомлению из почты и также - на экране своего личного кабинета системы №2. Далее отдел снабжения проверяет наличие запрашиваемого пользователем оборудования в системе №3.
Если оборудование есть - снабжение вызывает логистику и отправляет оборудование первой линии технической поддержки. Техническая поддержка выдаёт это оборудование пользователю и закрывает заявку пользователя в системе №1 и закрывает заявку в системе №2. Пользователь получает оборудование и может выставить оценку сервиса технической поддержке.
Если оборудования нет - снабжение делает заявку на закупку поставщику через почту. Ждёт оборудование. Поставщик его привозит. Далее снабжение проводит приём оборудования у себя в системе №3. Далее вызывает логистику и передаёт оборудование технической поддержке. Техническая поддержка выдаёт это оборудование пользователю и закрывает заявку пользователя в системе №1 и закрывает заявку в системе №2. Пользователь получает оборудование и может выставить оценку сервиса технической поддержке.

На модели можно указать вопросы, противоречия к описанию рабочего потока. (Чего не хватает для грамотного отражения модели «как есть”)

Задание 2:

Описание задачи:

Представьте, что вы аналитик Личного кабинета продавца.
Вам поступила задача сформулировать требования для реализации новой фичи (функциональности), которая позволит опубликовать свой товар на маркетплейсе (витрине товаров).

Перед тобой следующие задачи:

- *Сделать верхнеуровневую постановку в формате User story и описать требования к новой функциональности в формате вариантов использования (Use Case).*

- *Опционально: нарисовать диаграмму процесса в удобной нотации*

USER STORY

Как продавец,

Я хочу опубликовать свой товар на маркетплейсе,

Чтобы он стал виднее потенциальным покупателям

Примечания:

- обсудить с командой разработки, будет ли новая фича доступа и как отдельная функция, и как дополнительная функция при регистрации / создании нового товара продавцом

- будет ли доступна функция выбрать несколько товаров для публикации одновременно

- будет ли доступна функция выбрать все товары сразу для публикации

- уточнить системные этапы / статусы публикации: ‘на модерации’, ‘опубликован’, ‘отклонен’

- валидация: уточнить критерии, по которым товар может / не может быть опубликован; сделать инструкцию / подсказку для пользователя

- права доступа: публикация доступна всем пользователям?

- может ли продавец снять товар с публикации?

- время отклика системы? время процесса ‘на модерации’?

Критерии приемки:

1. Продавец опубликовал свой товар на маркетплейсе

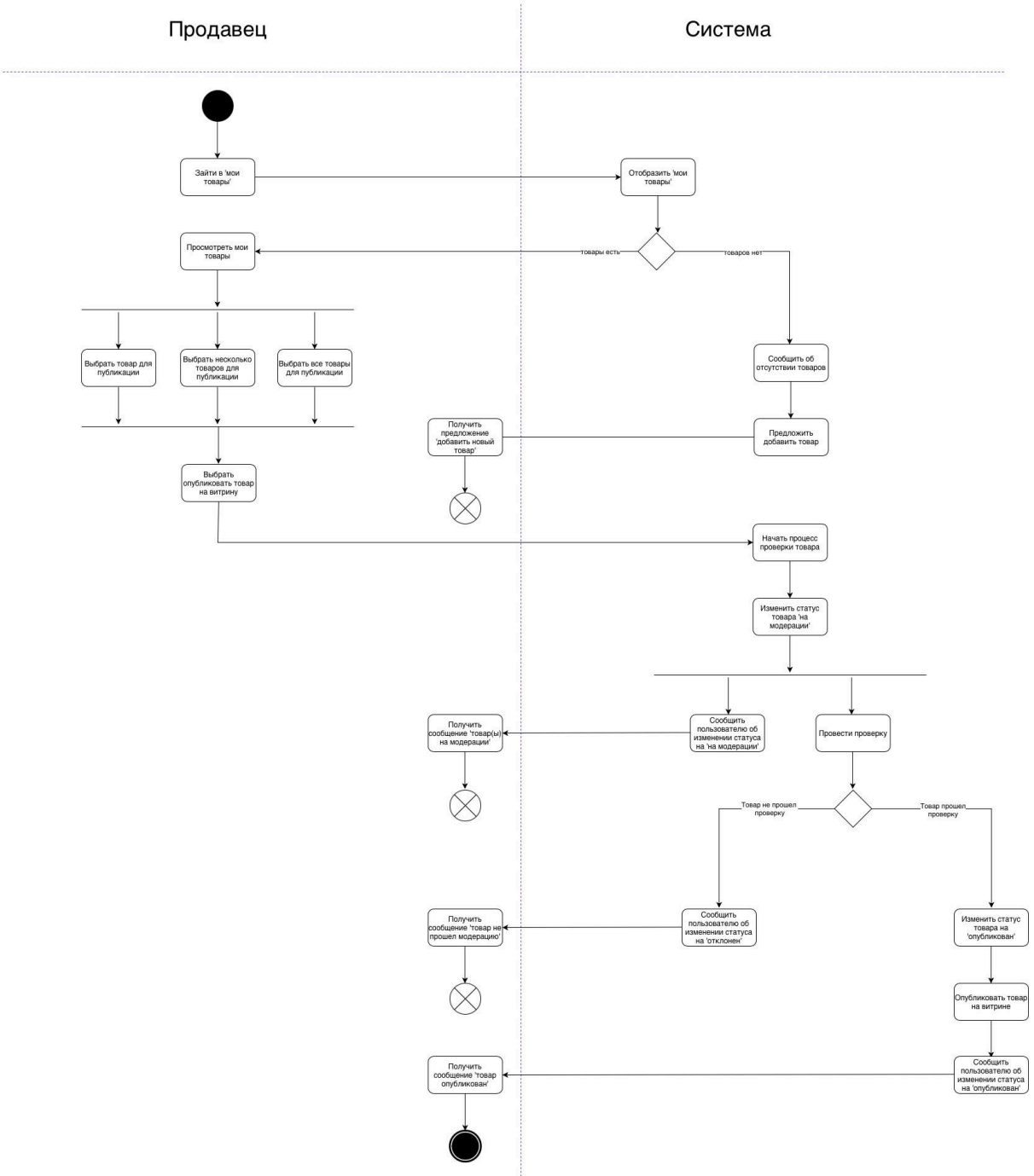
   Сценарий: Продавец опубликовал свой товар на маркетплейсе
   Дано: у продавца есть зарегистрированные товары
   Когда продавец просматривает свои зарегистрированные товары
   Тогда продавец может выбрать опубликовать отдельный товар на маркетплейсе

USE CASE

Уникальный код и название	**ВИ-1: ОПУБЛИКОВАТЬ СВОЙ ТОВАР НА МАРКЕТПЛЕЙСЕ**
Контекст использования	Добавление товара на витрину маркетплейса
Область действия	Маркетплейс XXX
Уровень	Цели пользователя
Основное действующее лицо	Продавец
Предусловие	Успешно выполнен базовый сценарий варианта использования “Зарегистрировать новый товар на маркетплейсе”
Минимальные гарантии успеха	Товар не опубликован
Гарантии успеха	Товар опубликован на маркетплейсе
Триггер	Продавец хочет разместить свой товар на витрине маркетплейса
Базовый сценарий	Опубликовать свой товар на витрине маркетплейса
1	Продавец открывает свои товары
2	Продавец выбирает товар, который он хочет опубликовать на маркетплейсе
3	Продавец подтверждает свой выбор
4	Система отображает продавцу статус ‘Публикация товара: на модерации’
5	Публикация товара проходит проверку системой
6	Статус товара меняется на ‘Опубликован’
7	Система отображает выбранный товар на витрине маркетплейса

Расширения	- У продавда не найдено товаров - Продавец хочет добавить несколько товаров одновременно на витрину - Продавец хочет добавить сразу все свои товары на витрину - Система не смогла добавить товар на витрину - Товар не прошел валидацию
1а	Система отображает продавцу сообщение о том, что добавленных товаров не найдено
1а1	Система предлагает продавцу варианты перехода: - добавить новый товар - назад на главное меню
2а	Продавец выбирает сразу несколько товаров для публикации на маркетплейсе
2а1	Продавец возвращается на шаг 3 основного сценария
2б	Продавец выбирает опубликовать сразу все товары на маркетплейсе
2б1	Продавец возвращается на шаг 3 основного сценария
4а	Система отображает продавцу сообщение об ошибке
4а1	Система предлагает продавцу опции: - попробовать снова - отменить публикацию
5а	Товар не прошел проверку системой
5а1	Система отсылает уведомление продавцу о статусе товара ‘Отклонен’
Изменения в технологии и данных	1. В таблице Products появляется новое поле ‘статус’ 2. Фиксируется дата публикации 3. Фиксируются изменения статусов

UML ACTIVITY DIAGRAM



Задача 3.1

Описание задачи:

На картинке (скачать можно по ссылке - https://drive.google.com/file/d/1fi9JLRvO6bhS1uJJ2PUX8ZTOq0Eq_At0/view?usp=drive_link) изображен интерфейс регистрации Пользователя (как успешный экран создания Пользователя, так и варианты с ошибками).

Задание 3.1:

Требуется описать REST api интерфейс на стороне бэк сервиса, который должен вызываться фронтендом при нажатии на кнопку "Register". Описание должно включать:

- *http метод + URL*
- *входные параметры (наименование, тип, обязательность, ограничения)*
- *выходные параметры при успешном ответе (наименование, тип, обязательность, ограничения)*
- *выходные параметры при ответе с ошибкой (наименование, тип, обязательность, ограничения)*
- *описание ошибок - коды ответов (главное верно указать клиентские или серверные) + текст сообщения*
- *пример запроса*
- *пример ответа (для успешного сценария и сценария с ошибкой).*

Требований к формату нет (может быть в виде таблицы или просто текстом).

Допускается описание в формате openAPI спецификации. Полученный yaml приложить в виде текста к решению или в виде отдельного файла (обязательно не забыть открыть доступ).

openAPI:

*прикладываю yaml в виде текста ниже
**также в репозитории есть отдельный файл задание3.1.yaml, если это будет удобнее

```
openapi: 3.0.0

info:

  title: Book Store Registration API

  description: API для регистрации пользователя в сервисе Book Store

  version: 1.0.1

  contact:

    name: Registration API

    url: https://api.demoga.com/register

    email: registration@demoga.com

  termsOfService: https://demoga.com/terms
```

```
  license:
    name: license 1.0.1 Book Store
    url: https://demoga.com/license

servers:
  - url: https://api.demoga.com
    description: Production server

tags:
  - name: Registration

paths:
  /register:
    post:
      tags:
        - Registration
      summary: Регистрация нового пользователя
      description: >
        Создание нового аккаунта пользователя.
        При успешной регистрации фронтенд должен показать модальное окно
        с текстом из поля `message`.
      operationId: registerUser

      requestBody:
        required: true
        content:
          application/json:
```

```
        schema:
          $ref: '#/components/schemas/RegisterRequest'

    responses:
      '201':
        description: Пользователь успешно зарегистрирован
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/RegisterSuccessResponse'
            example:
              userId: "550e8400-e29b-41d4-a716-446655440000"
              username: "ivan"
              message: "User Register Successfully"

      '400':
        description: Некорректный формат запроса (отсутствуют обязательные
поля)
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorResponse'
            example:
              errorCode: "BAD_REQUEST"
              message: "Required fields are missing."

      '409':
```

```
        description: Пользователь с таким username уже существует
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorResponse'
            example:
              errorCode: "USER_ALREADY_EXISTS"
              message: "User exists!"

      '422':
        description: Ошибка бизнес-валидации (слабый пароль или не пройдена капча)
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorResponse'
            examples:
              weakPassword:
                value:
                  errorCode: "WEAK_PASSWORD"
                  message: "Passwords must have at east one non alphanumeric character, one digit (0-9), one uppercase (A-Z), one lowercase (a-z), one special character and Password must be eight characters or longer.Passwords must have at east one non alphanumeric character, one digit (0-9), one uppercase (A-Z), one lowercase (a-z), one special character and Password must be eight characters or longer."
              captchaFailed:
                value:
                  errorCode: "CAPTCHA_VALIDATION_FAILED"
                  message: "Please verify reCaptcha to register!"
```

```
        '500':
          description: Внутренняя ошибка сервера
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
              example:
                errorCode: "INTERNAL_SERVER_ERROR"
                message: "Oops, something went wrong. Please try again later."

components:
  schemas:

    RegisterRequest:
      type: object
      required:
        - firstName
        - lastName
        - username
        - password
        - captchaToken
      properties:
        firstName:
          type: string
          minLength: 2
          maxLength: 50
```

```
    example: ivan

  lastName:
    type: string
    minLength: 2
    maxLength: 50
    example: ivanov

  username:
    type: string
    minLength: 4
    maxLength: 30
    pattern: "^[a-zA-Z0-9_]+$"
    example: ivan

  password:
    type: string
    minLength: 8
    maxLength: 100
    format: password
    example: StrongPass123!

  captchaToken:
    type: string
    description: Токен подтверждения reCAPTCHA
    example: 03AFcWeA...
```

```
RegisterSuccessResponse:
  type: object
  required:
    - userId
    - username
    - message
  properties:
    userId:
      type: string
      format: uuid
      example: 550e8400-e29b-41d4-a716-446655440000

    username:
      type: string
      example: ivan

    message:
      type: string
      description: >
        Сообщение, которое фронтенд должен показать пользователю
        во всплывающем окне.
      example: User Register Successfully

ErrorResponse:
  type: object
  required:
    - errorCode
```

```
    - message
  properties:
    errorCode:
      type: string
      description: Машиночитаемый код ошибки
      example: USER_ALREADY_EXISTS
    message:
      type: string
      description: Человекочитаемое описание ошибки
      example: User exists!
```

Задание 3.2

Смотри отдельный jpg-файл ‘задание3.2’ в репозитории.

Описание задачи:

Описать подробный пошаговый алгоритм, создания Пользователя на стороне бэк сервиса при вызове, получившегося в Задачи 3.1 метода. Алгоритм должен подробно описывать какие проверки требуется выполнять бэк сервису после получения запроса на создание нового Пользователя (при нажатии кнопки “Register”), прежде чем создать Пользователя в БД. Описание может быть в виде текста или диаграммы (например, activity)/блок-схемы.