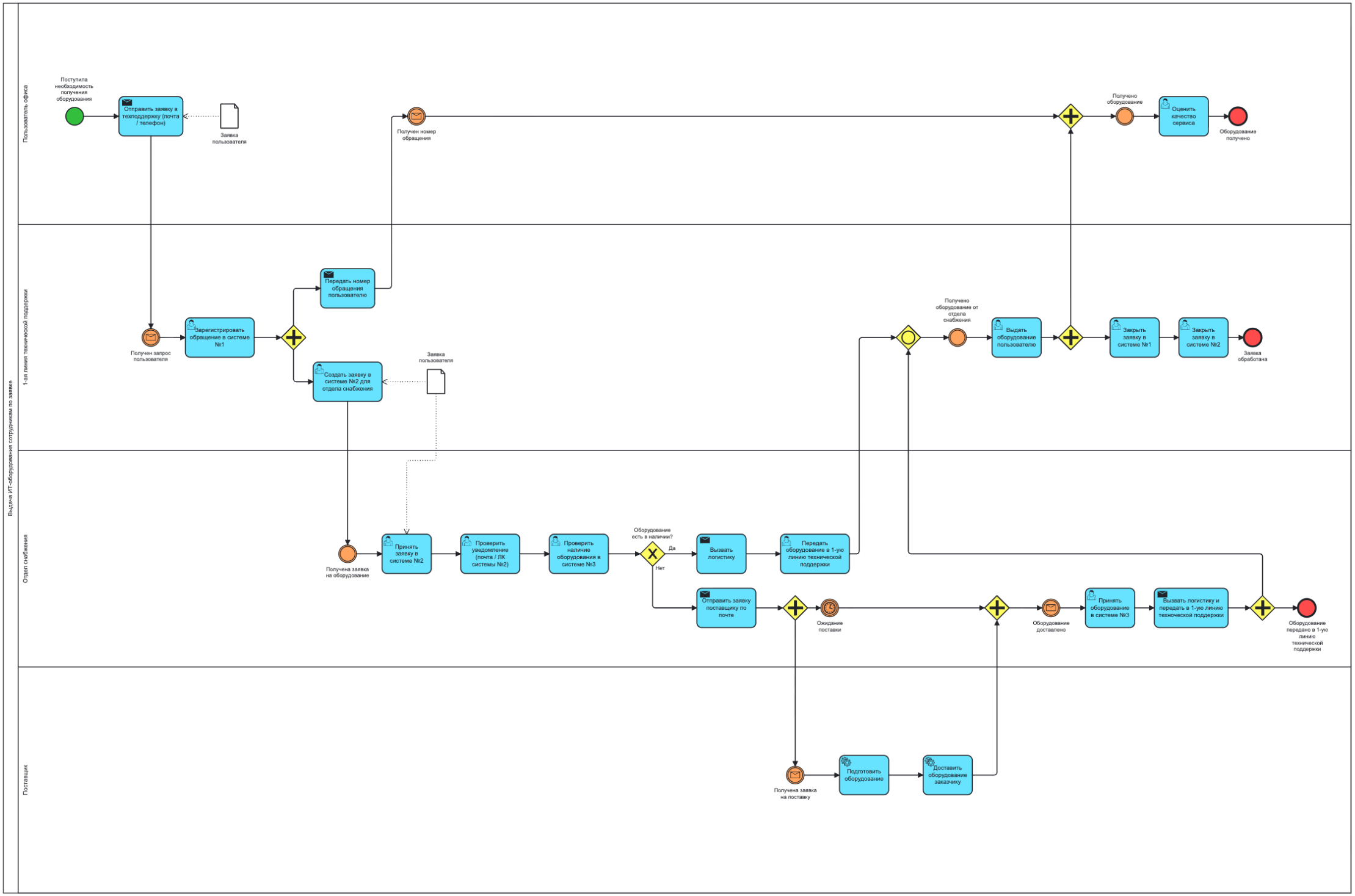
# Техническое задание (БА+СА) аналитик

**Задача 1**

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

Противоречия:

1. Логистика не выделена как отдельный участник процесса
2. Нет регламентов сроков (SLA) – нет таймеров ожидания поставки и реакций подразделений
3. Не определено, кто уведомляет пользователя о готовности оборудования
4. Закрытия заявок через систему №1 и №2 выполняются вручную – требуется автоматизация


Выдача ИТ-оборудования сотрудникам по заявке
Пользователь офиса
1-ая линия технической поддержки
Отдел снабжения
Поставщик
Поступила необходимость получения оборудования
Отправить заявку в техподдержку (почта / телефон)
Заявка пользователя
Получен номер обращения
Получено оборудование
Оценить качество сервиса
Оборудование получено
Получен запрос пользователя
Зарегистрировать обращение в системе №1
Передать номер обращения пользователю
Создать заявку в системе №2 для отдела снабжения
Заявка пользователя
Получено оборудование от отдела снабжения
Выдать оборудование пользователю
Закрыть заявку в системе №1
Закрыть заявку в системе №2
Заявка обработана
Получена заявка на оборудование
Принять заявку в системе №2
Проверить уведомление (почта / ЛК системы №2)
Проверить наличие оборудования в системе №3
Оборудование есть в наличии?
Да
Нет
Вызвать логистику
Передать оборудование в 1-ую линию технической поддержки
Отправить заявку поставщику по почте
Ожидание поставки
Оборудование доставлено
Принять оборудование в системе №3
Вызвать логистику и передать в 1-ую линию техноческой поддержки
Оборудование передано в 1-ую линию технической поддержки
Получена заявка на поставку
Подготовить оборудование
Доставить оборудование заказчику

**Задача 2**

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

2. Опционально: нарисовать диаграмму процесса в удобной нотации.

**User story:**

**Как продавец** маркетплейса,

**я хочу** иметь возможность опубликовать свой товар через Личный кабинет,

**чтобы** мой товар появился на витрине маркетплейса и стал доступен пользователям

**Acceptance Criteria:**

- Продавец может создать и отредактировать карточку товара
- Перед публикацией данные проходят валидацию
- Товар отправляется на модерацию
- Продавец видит статус товара
- Продавец получает результат модерации (одобрен/отклонен)

**Use Case**

**UC1. Создание карточки товара**

- **Актор:** продавец
- **Описание:** Продавец создаёт карточку товара и заполняет основные характеристики
- **Результат:** Карточка сохранена со статусом «Черновик»

**UC2. Редактирование карточки товара**

- **Актор:** Продавец
- **Описание:** Продавец вносит изменения в ранее созданную карточку
- **Результат:** Обновлённые данные сохранены

**UC3. Отправка товара на модерацию**

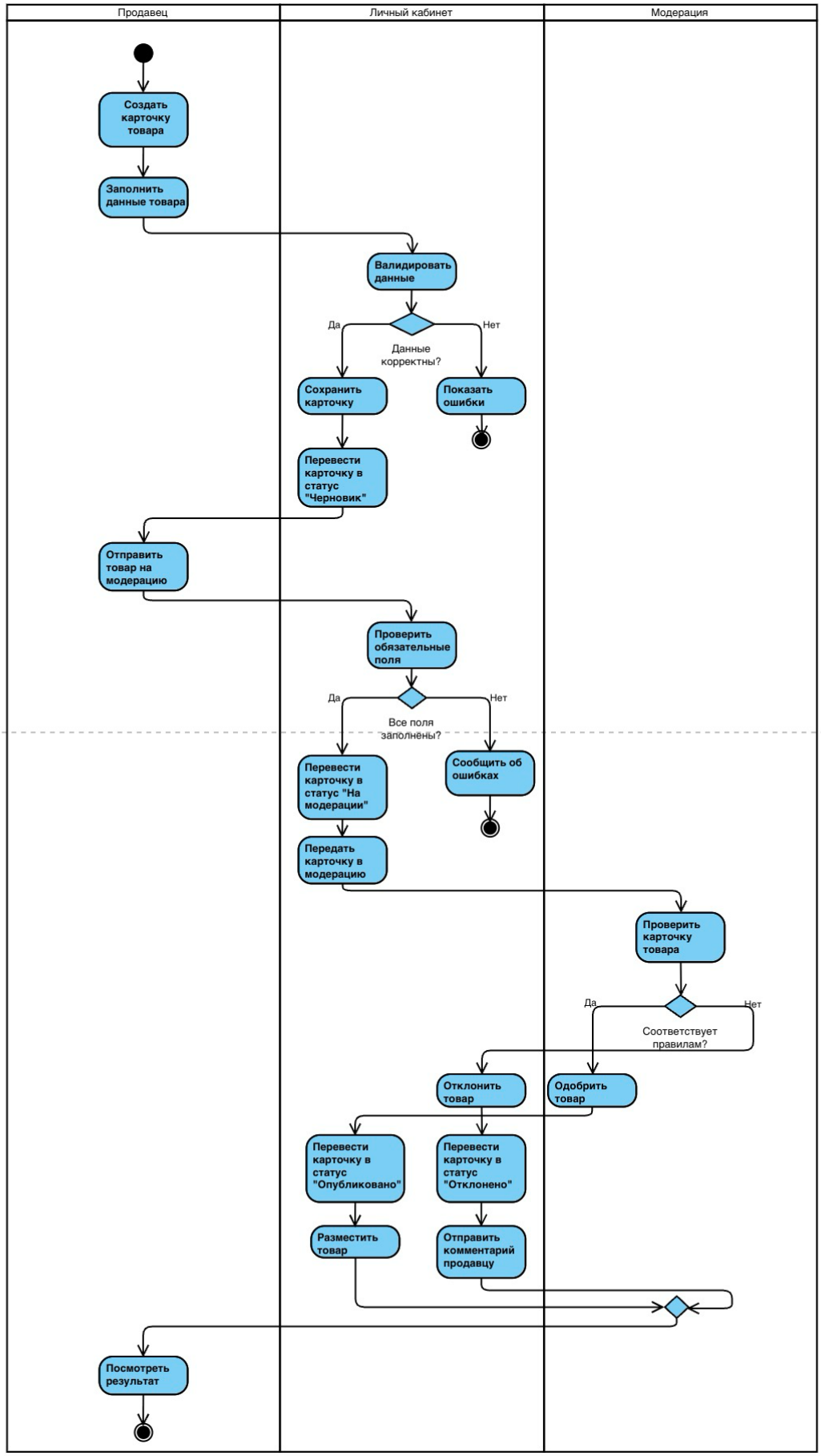
- **Актор:** Продавец
- **Предусловие:** Все обязательные поля заполнены
- **Описание:** Продавец отправляет карточку товара на проверку
- **Результат:** Статус товара — «На модерации»

**UC4. Модерация товара**

- **Актор:** Модератор маркетплейса
- **Описание:** Проверка карточки товара на соответствие правилам
- **Альтернативы:**
  - Одобрение товара
  - Отклонение товара с комментарием
- **Результат:** Обновлён статус товар

**UC5. Публикация товара**

- **Актор:** Система
- **Описание:** При успешной модерации товар публикуется на витрине
- **Результат:** Товар доступен покупателям


Продавец
Личный кабинет
Модерация
Создать карточку товара
Заполнить данные товара
Валидировать данные
Данные корректны?
Да
Нет
Сохранить карточку
Показать ошибки
Перевести карточку в статус "Черновик"
Отправить товар на модерацию
Проверить обязательные поля
Все поля заполнены?
Да
Нет
Перевести карточку в статус "На модерации"
Сообщить об ошибках
Передать карточку в модерацию
Проверить карточку товара
Соответствует правилам?
Да
Нет
Отклонить товар
Одобрить товар
Перевести карточку в статус "Опубликовано"
Перевести карточку в статус "Отклонено"
Разместить товар
Отправить комментарий продавцу
Посмотреть результат

Задача 3

Задание 3.1:

Требуется описать REST api интерфейс на стороне бэк сервиса, который должен вызываться фронтендом при нажатии на кнопку “Register”

- **HTTP метод:** POST
- **URL:** /api/v1/users/register
- **Назначение:** регистрация нового пользователя
- **Формат данных:** application/json

```yaml
openapi: 3.0.3
info:
  title: User Registration API
  description: API для регистрации нового пользователя
  version: 1.0.0

paths:
  /api/v1/users/register:
    post:
      summary: Регистрация нового пользователя
      operationId: registerUser
      tags:
        - Users
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/UserRegistrationRequest'
      responses:
        '201':
          description: Пользователь успешно зарегистрирован
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/UserRegistrationResponse'
        '400':
          description: Ошибка валидации данных
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ErrorResponse'
        '409':
          description: Пользователь с таким username уже существует
          content:
            application/json:
              schema:
```

```yaml
              $ref: '#/components/schemas/ErrorResponse'
      '500':
        description: Внутренняя ошибка сервера
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ErrorResponse'

components:
  schemas:
    UserRegistrationRequest:
      type: object
      required:
        - firstName
        - lastName
        - username
        - password
        - captchaToken
      properties:
        firstName:
          type: string
          example: Ivan
        lastName:
          type: string
          example: Ivanov
        username:
          type: string
          example: ivan_ivanov
        password:
          type: string
          format: password
          example: Qwerty!23
        captchaToken:
          type: string
          description: Токен reCAPTCHA от фронтенда
          example: 03AF6jDq...

    UserRegistrationResponse:
      type: object
      properties:
        userId:
          type: string
          example: "a12f45c9-9e3a-4c12-8a5b-91d8f123abcd"
        message:
          type: string
          example: User registered successfully

    ErrorResponse:
      type: object
      properties:
        errorCode:
          type: string
          example: VALIDATION_ERROR
```

```
        message:
          type: string
          example: Password does not meet complexity requirements
```

Задание 3.2

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.

1. Получение запроса

Backend получает HTTP POST запрос с телом:

- firstName
- lastName
- username
- password
- captchaToken

2. Техническая валидация запроса

Backend получает HTTP POST запрос с телом:

- firstName
- lastName
- username
- password
- captchaToken

Если возникает ошибка -> 400 Bad Request

3. Валидация CAPTCHA
    - Отправить captchaToken во внешний сервис reCAPTCHA
    - Проверить:
        - Валидность токена
        - Запрос не бот

Если CAPTCHA не пройдена -> 400 Bad Request “Captcha validation failed”

4. Валидация бизнес-правил полей

- Имя и фамилия:
  - не пустые
  - длина (например, 2–50 символов)
  - допустимые символы (буквы, дефис)
- Username:
  - не пустой
  - соответствует формату (латиница, цифры, _)
  - длина допустима
- Password:
  - не менее 8 символов
  - содержит:
    - минимум 1 цифру
    - минимум 1 заглавную букву
    - минимум 1 строчную букву
    - минимум 1 спецсимвол (!@#$%^&*)

Если хотя бы одна проверка не пройдена -> 400 Bad Request (в сообщении аналогично отображаемому на UI)

5. Проверка уникальности пользователя

Поиск пользователя в БД по username, в случае ошибки (Пользователь уже существует) -> 409 Conflict “User already exists”

6. Хэширование пароля

Захешировать пароль, никогда не сохранять пароль в открытом виде

7. Создание пользователя в БД

Создать запись пользователя:

- id
- first_name
- last_name
- username

- password_hash
- created_at
- status = ACTIVE

8. Формирование ответа

Вернуть 201 Created, userId, сообщение об успешной генерации.

9. Можно добавит логирование события регистрации, отправку события в аналитическую систему для отслеживания метрик.