Оглавление

[1) Создание базовой лоукодной модели (обязательное) 2](#_Toc167141490)

[2) Описание структуры бизнес-процесса\* (опциональное) (аналитика) 3](#_Toc167141491)

[3) Менеджмент ресурсов (обязательное) 5](#_Toc167141492)

[4) Какие сетевые условия надо соблюсти в рамках тестирования (куда задеплоить компонент, откуда к нему обращаться), чтобы гарантировать валидность проверки яндекс капчи. Какую ценность предоставляет капча для вашей системы и на что как ПМ вы бы обратили внимание при принятии результата по тест плану от тестировщиков?" 7](#_Toc167141493)

# 1) Создание базовой лоукодной модели (обязательное)

Предложите учебно-демонстрационную дата-модель простой лоу-код системы для управления поручениями и\или проектами и обоснуйте, почему ваша датамодель имеет именно такой набор сущностей\связей\атрибутов. В чем преимущества вашего приложения по сравнению с хардкодным аналогом? За основу вы можете взять системы Monday.com, trello.com, JIRA.

Дата-модель может быть представлена в виде диаграммы "окошечно-облачного" типа, таблицы etc., используйте любой инструмент, который наиболее точно, по вашему мнению, покажет как именно выглядит модель данных вашего лоу-код приложения.

**Учебно-демонстрационная Дата-Модель**

**1. Сущности:**

* **Проект (Project):**
  + Атрибуты: ID проекта, Название проекта, Описание, Статус проекта (активный, завершенный и т. д.), Связь с другим проектом в виде ссылки (если существует).
* **Задача (Task):**
  + Атрибуты: ID задачи, Название задачи, Описание, Дата начала, Дата завершения, Приоритет (низкий, средний, высокий), Статус задачи (открыта, в работе, завершена и т. д.), Связь с проектом (ссылка на источник, например, git).
* **Исполнитель (Assignee):**
  + Атрибуты: ID исполнителя, Имя, Электронная почта, Роль (менеджер, разработчик, аналитик).

**2. Связи:**

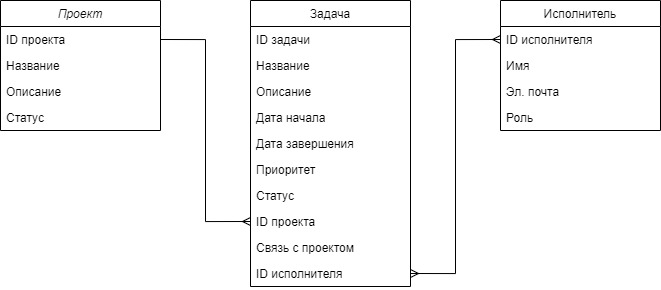
* **Проект - Задача (Project-Task):**
  + Один проект может иметь много задач.
  + Каждая задача принадлежит только одному проекту.
* **Исполнитель - Задача (Assignee-Task):**
  + Один исполнитель может быть назначен на несколько задач.
  + Одна задача может иметь нескольких исполнителей.

**3. Примеры действий:**

* Создание нового проекта.
* Добавление новой задачи в проект.
* Назначение исполнителей на задачу.
* Обновление статуса задачи.
* Просмотр списка задач для определенного проекта.

**Преимущества по сравнению с хардкодным аналогом**

* **Гибкость и Пользовательский опыт:** Лоу-код система позволяет пользователям быстро настраивать и адаптировать систему под свои нужды без необходимости программирования. Это делает приложение более доступным и удобным для широкого круга пользователей.
* **Быстрое развертывание:** В отличие от хардкодных систем, которые могут требовать значительного времени и ресурсов на разработку и тестирование, лоу-код система на основе данной модели может быть быстро развернута и готова к использованию.
* **Легкость адаптации:** Лоу-код система позволяет легко вносить изменения в функциональность и структуру приложения в соответствии с изменяющимися потребностями бизнеса или проекта, что делает ее более гибкой и адаптивной.



# 2) Описание структуры бизнес-процесса\* (опциональное) (аналитика)

Представьте, что закрытие задач определенного типа в вашем приложении требует аппрува от "проверяющего" + заказчика (с использованием электронной подписи).

Пример:

Фронтенд-разработчик Вася получил задачу по перекрашиванию кнопки из зеленого цвета в синий. Поверяющим задачи является Петя - тимлид Васи. Заказчиком изменения - Дмитрий, оунер продукта.

Чтобы задача оказалась закрыта и Вася получил оплату за его выполнение, задача должна быть проверена Петей, заапрувлена Дмитрием (с использованием электронной подписи), затем информация должна дойти до бухгалтерии, где Васе оформят денежный перевод.

Опишите структуру бизнес-процесса (разбиение на этапы, с указанием Process Input и Process Output каждого этапа) по аппруву результатов задачи. Предложите ролевую модель для участников бизнес-процесса.

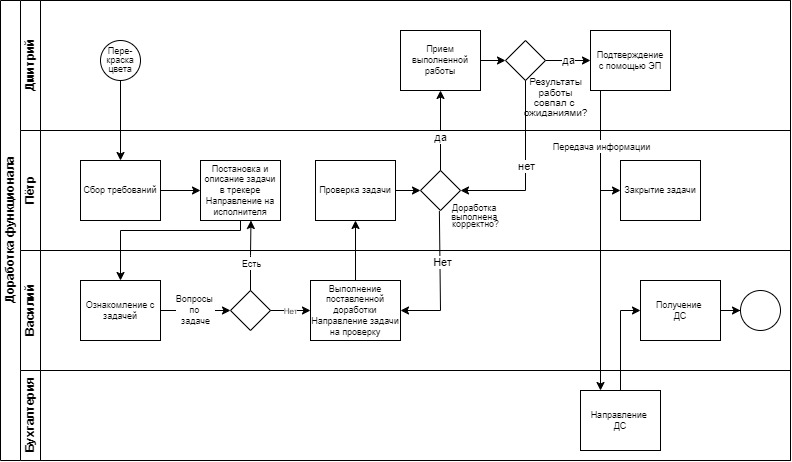
Ответ принимается в виде текстового документа любого вида, который пошагово описывает процесс + BPMN диаграммы, которая его визуализирует, также предоставляя сведения о разбивке процесса по ролевой модели.

Примечание:

Для описания бизнес-процесса можно использовать обычный русский язык, который описывает процесса в виде ПОСЛЕДОВАТЕЛЬНОСТИ элементов BPMN 2.0, при этом на ход выполнения процесса могут влиять как внешние СОБЫТИЯ, так и результаты выполнения конкретных операций (шлюзы).

**Структура бизнес-процесса:**

1. **Поучение запроса от заказчика:**
   * *Process Input:* Запрос заказчика.
   * *Process Output:* Оформленная задача в таск-трекере направленная на разработку.
2. **Получение задачи от тимлида:**
   * *Process Input:* Поставленная задача в таск-трекере.
   * *Process Output:* Выполненная задача, переданная на проверку.
3. **Получение завершенной задачи от разработчика:**
   * *Process Input:* Завершенная задача от фронтенд-разработчика.
   * *Process Output:* Решение о дальнейших действиях (аппрув или возврат задачи на доработку).
4. **Аппрув заказчиком (владельцем продукта):**
   * *Process Input:* Задача, прошедшая проверку тимлида.
   * *Process Output:* Окончательное решение (аппрув или возврат задачи), передача информации в бухгалтерию.
5. **Подтверждение и оплата:**
   * *Process Input:* Задача, одобренная заказчиком.
   * *Process Output:* Отправка оплаты.



# 3) Менеджмент ресурсов (обязательное)

Представьте, что в последний момент перед релизом (3 дня до даты релиза) заказчик внезапно изменил требование к приложению, которое разрабатывало ваша команда. В веб-компоненте вашего лоукод-приложения по управлению проектами требуется поменять captcha компонент, чтобы использовалась yandex-captcha вместо google-captcha. Компания пошла навстречу заказчику и требует, чтобы вы выполнили задачу. Какие ресурсы вы запросите, как вы их распределите и какие задачи поставите для выполнения ассайнмента и достижения цели? Как вы удостоверитесь, что несмотря на сжатые сроки весь остальной функционал приложения не пострадал?

Изменение требований так близко к дате релиза может представлять определенные вызовы, но с правильным планированием и организацией задач можно успешно внести необходимые изменения, минимизируя риски для остального функционала приложения.

**1. Оценка задачи:**

* **Разбор требований**: Понять, какой функционал требуется изменить (замена Google reCAPTCHA на Yandex reCAPTCHA).
* **Технический анализ**: Оценить объем работы, необходимые изменения в коде, потенциальные зависимости и риски.
* **Оценка времени**: Определить точные временные рамки для выполнения задачи в короткий срок.

**2. Распределение ресурсов:**

* **Выделение разработчика**: Назначить опытного разработчика, знакомого с компонентами reCAPTCHA и интеграцией в веб-приложения.
* **Поддержка QA**: Задействовать QA-специалиста для тестирования изменений и обеспечения качества.

**3. План действий:**

* **Интеграция Yandex reCAPTCHA**:
  + Разработчик должен интегрировать новый captcha компонент в веб-компонент проекта.
  + Провести тестирование, чтобы убедиться, что новый captcha работает корректно и безопасно.

**4. Контроль качества и управление рисками:**

* **Тестирование и отладка**: QA-специалист должен провести тщательное тестирование нового captcha и убедиться, что остальной функционал приложения остался неповрежденным.
* **Обратная связь с заказчиком**: Демонстрация и получение обратной связи от заказчика по новой реализации captcha.

**5. Мониторинг и завершение:**

* **Мониторинг в процессе**: Регулярное отслеживание прогресса и обмен информацией между разработчиком и QA-специалистом.
* **Завершение и отчет**: Подготовка и представление отчета о выполненной работе и результате тестирования.

**6. Контроль изменений:**

* **Управление версиями**: Использование системы контроля версий (например, Git) для отслеживания изменений и восстановления предыдущего состояния при необходимости.

**Обеспечение безопасности и качества:**

* **Безопасность**: Гарантировать безопасность нового captcha компонента и соответствие принципам защиты данных.
* **Качество**: Убедиться, что функционал приложения работает стабильно и соответствует ожиданиям пользователя.

При выполнении этой задачи ключевым является четкое планирование, прозрачная коммуникация и тесное сотрудничество между разработчиком, QA-специалистом и заказчиком. Должна быть также готовность к быстрым реакциям на любые непредвиденные проблемы или задержки.

# 4) Какие сетевые условия надо соблюсти в рамках тестирования (куда задеплоить компонент, откуда к нему обращаться), чтобы гарантировать валидность проверки яндекс капчи. Какую ценность предоставляет капча для вашей системы и на что как ПМ вы бы обратили внимание при принятии результата по тест плану от тестировщиков?"

1.

1) Доработанный (изменённый) вариант развернуть на тестовом сервере, доступным для https и проверить на требования от яндекс капчи

2) зайти на сайт яндекс капчи для получения апи ключей (site key, secret key) и внедрить в проект (вставить html код где-то в форме регистрации, например). Так же на сервере добавить обработчик (например, на Flask). Нашла пример:

@app.route('/submit, methods=['POST'])

class Config:

SECRET\_KEY = os.getenv('SECRET\_KEY', 'your\_secret\_key')

YANDEX\_CAPTCHA\_SITE\_KEY = os.getenv('YANDEX\_CAPTCHA\_SITE\_KEY', 'your\_site\_key')

YANDEX\_CAPTCHA\_SECRET\_KEY = os.getenv('YANDEX\_CAPTCHA\_SECRET\_KEY', 'your\_secret\_key')

2.

1) Капча проверяет на реалистичность пользователя (что он человек, а не программа), помогает предотвратить автоматические атаки (скан, спам, подбор паролей, взлом и тд) и сохранить данные пользователей

2) Помогает снизить нагрузку на серверы и бд

3.

1) Проверить во все ли нужные формы внедрена капча, вдруг что-то забыли

2) Самая главная проверка, на мой взгляд, нагрузочного тестирования. Проверить автоматизированные запросы и обходные варианты атак.

3) Проверить не повлечет ли за собой неприятности для пользователя, не создает ли она сложностей для пользования (например, долгие прогрузки системы).

4) Пишутся ли логи событий.