Тестовое задание на позицию .NET Developer: Разработка REST API для системы квестов в игре

Описание задачи

Необходимо разработать REST API для системы управления квестами в игре. Система должна позволять игрокам:

- Просматривать доступные квесты.
- Принимать квесты.
- Обновлять прогресс выполнения квестов.
- Завершать квесты и получать награду.

Функциональные требования

1. Основные сущности системы:

- **Игрок:**
    - Уникальный идентификатор игрока.
    - Информация об игроке (имя, уровень)
- **Квест:**
    - Уникальный идентификатор квеста.
    - Название и описание квеста.
    - Условия выполнения квеста (собрать определенные предметы, победить определенных монстров, посетить определенную локацию).
    - Награды за выполнение квеста (опыт, предметы, валюта).
    - Требования для доступа к квесту (минимальный уровень игрока, предыдущие выполненные квесты).
- **Квест игрока:**
    - Уникальный идентификатор игрока квеста игрока.
    - Статус квеста для игрока**:**
        - **Принят (Accepted):** игрок принял квест и начал его выполнение.
        - **В процессе (In Progress):** квест активен, игрок выполняет его условия.
        - **Выполнен (Completed):** все условия квеста выполнены, но награда еще не получена.

        - **Завершен (Finished):** игрок получил награду, квест полностью завершен.
    - Прогресс выполнения условий квеста для игрока.

2. Основные функции системы:

2.1. Просмотр доступных квестов:

- **Описание:**
    - Позволить игроку получать список квестов, доступных для принятия.
- **Требования:**
    - Система должна предоставлять только те квесты, которые доступны данному игроку с учетом его уровня и ранее выполненных квестов.
    - Квесты, которые уже приняты или завершены игроком, не должны отображаться в списке доступных.

2.2. Принятие квеста:

- **Описание:**
    - Позволить игроку принять выбранный квест.
- **Требования:**
    - Перед принятием квеста система должна проверять:
        - Доступен ли квест для данного игрока.
        - Не превышено ли максимальное количество активных квестов у игрока (максимум 10 активных квестов).
    - После успешного принятия квеста должна создаваться запись о прогрессе выполнения этого квеста для игрока, при этом статус квеста должен оставаться “Принят” до тех пор пока не будет какого либо прогресса по нему.

2.3. Обновление прогресса выполнения квеста:

- **Описание:**
    - Позволить игроку обновлять прогресс выполнения принятого квеста.
- **Требования:**
    - Система должна корректно обновлять только те параметры прогресса, которые были изменены.
    - Обеспечить возможность частичного обновления прогресса квеста напрямую через один эндпоинт **(partial update),** либо косвенно, как реакция на определенные события игрока**.**
    - Если квест уже находится в статусе “Выполнен” или “Завершен”, то никаких проверок проводить не нужно, просто возвращать текущий прогресс.
    - При обновлении прогресса система должна проверять:

        - Что значения прогресса не уменьшаются.
        - Корректность новых значений (например, прогресс по определенному условию квеста не может превышать требуемое значение).
        - Если квест находился в статусе “Принят”, то статус должен измениться на “В процессе”.
    - Если условия квеста выполнены:
        - Статус квеста должен автоматически измениться на "Выполнен".
    - Информация о текущем прогрессе должна возвращаться в ответе на обновление прогресса.

2.4. Завершение квеста и получение награды:

- **Описание:**
    - Позволить игроку завершить квест и получить соответствующую награду.
- **Требования:**
    - Если квест находится в статусе “Завершен”, то система просто информирует об этом пользователя.
    - Перед завершением квеста система проверяет, что текущий статус квеста "Выполнен".
        - Если условие выполнены:
            - Игрок получает соответствующую награду.
            - Обеспечить, чтобы награда выдавалась только один раз за квест.
            - Статус квеста обновляется на "Завершен".
        - Если условие не выполнено:
            - Система информирует игрока о том, какие условия квеста еще не выполнены.
            - Квест остается в текущем статусе.

3. Валидация и обработка ошибок:

- **Валидация данных:**
    - Проверять корректность и полноту всех входящих данных в запросах.
    - Обеспечивать соответствие данных заданным форматам и ограничениям (например, типы данных, диапазоны значений).
- **Обработка ошибок:**
    - Предоставлять понятные и информативные сообщения об ошибках при некорректных запросах или сбоях.
    - Обрабатывать исключительные ситуации внутри системы, не допуская неконтролируемых сбоев.

Технические требования

1. **Технологии:**
    - .NET 8.0
    - ASP.NET Core
    - EF Core
    - Docker для контейнеризации
    - Система должна быть реализована на базе Clean Architecture и DDD
2. **База данных:**
    - Любую другую подходящую базу данных
    - Инициализация базы данных при запуске приложения с данными
    - Возможно потребуется Docker Compose для разворачивание отдельного контейнера для базы и для приложения
3. **Тестирование:**
    - Написать юнит-тесты для слоя Domain с использованием xUnit или NUnit

Дополнительные требования

1. **Документация:**
    - Предоставьте краткое описание проекта и инструкции по его запуску в файле README.md.
2. **Качество кода:**
    - Придерживайтесь лучших практик написания кода (SOLID, KISS, DRY), уделяйте внимание читаемости и поддерживаемости.
3. **Тестовое покрытие:**
    - Обеспечьте адекватное покрытие тестами ключевых частей бизнес-логики.

Оценка задания

Критерии оценки будут включать:

- Соответствие техническому заданию.
- Качество и структура кода.
- Общая архитектура и использование лучших практик разработки.
- Уровень покрытия тестами и их качество.
- Внимание к деталям и обоснованность предложенных решений.

Ожидаемое время выполнения задания

Ожидаемое время выполнения задания: 5-7 дней.

Инструкции для отправки

1. Загрузите исходный код проекта на GitHub или другую платформу для совместной разработки.
2. Предоставьте ссылку на репозиторий.
3. Добавьте файл README.md с инструкциями по запуску приложения и тестов.