Конструирование программного обеспечения

Домашнее задание №2.
Закрепление концепции Domain-Driven Design и принципов проектирования Clean Architecture.

Дедлайн сдачи: 20 апреля, 23:59
Формат сдачи ДЗ определяется семинаристом. Задание выполняется на языке программирования являющимся основным для семинарских занятий (*иной язык только по согласованию с семинаристом*)

Предисловие

Московскому зоопарку вновь нужна ваша помощь!
Ранее разработанная вами информационная система оставила двоякое впечатление. С одной стороны, животные теперь всегда сыты, но не благодаря внедренной системе учета запасов, а из-за того, что информационная система не контролирует процесс расселения животных по вольерам. Как следствие, травоядных размещают к хищникам... Склады ломятся от запасов еды, а травоядные на гране исчезновения.
Скорее пиши код, останови этот ужас!

Вам предстоит разработать веб-приложение для автоматизации следующих бизнес-процессов зоопарка: управление животными; вольерами; расписанием кормлений.

В ходе общения с заказчиком были выявлены следующие требования к функциональности проектируемого модуля веб-приложения

Use Cases:

a. Добавить / удалить животное
b. Добавить / удалить вольер
c. Переместить животное между вольерами
d. Просмотреть расписание кормления
e. Добавить новое кормление в расписание
f. Просмотреть статистику зоопарка (кол-во животных, свободные вольеры и т.д.).

После встречи с доменными экспертами.

Вы определили три основных класса: Animal, Enclosure и FeedingSchedule. Архитектор проекта предложил вам следующий вариант создания богатой модели предметной области:

- Животное (Animal)
  - Вид, кличка, дата рождения, пол, любимая еда, статус (здоров/болен)
  - Методы: кормить, лечить, переместить в другой вольер
- Вольер (Enclosure)
  - Тип (для хищников, травоядных, птиц, аквариум и т.д.), размер, текущее количество животных, максимальная вместимость
  - Методы: добавить животное, убрать животное, провести уборку
- Расписание кормления (FeedingSchedule)
  - Животное, время кормления, тип пищи
  - Методы: изменить расписание, отметить выполнение

Выявление Value Object оставили на ваше усмотрение.

На kickoff митинге с командой было приняли решение, в первую очередь, реализовать следующие сервисы:

- AnimalTransferService (для перемещения животных между вольерами)
- FeedingOrganizationService (для организации процесса кормления)
- ZooStatisticsService (для сбора статистики по зоопарку)

Определить и реализовать следующие доменные события:

- AnimalMovedEvent (при перемещении животного)
- FeedingTimeEvent (при наступлении времени кормления)

Определили следующую структуру проекта, согласно Clean Architecture:

- Domain (ядро, содержит наши модели)
- Application (содержит сервисы, реализующие бизнес-логику приложения)
- Infrastructure (внешние взаимодействия)
- Presentation (контроллеры нашего веб-приложения)

ТРЕБУЕТСЯ

1. Разработать классы доменной модели согласно концепции Domain-Driven Design
2. Построить структуру проекта соблюдая принципы Clean Architecture.
3. В слое представления (Web API в архитектурном стиле REST API) реализовать следующие контроллеры:
    a. Контроллер для работы с животными (просмотр информации о животных, добавление новых, удаление);
    b. Контроллер для работы с вольерами;
    c. …
4. При помощи инструмента проектирования API (например Swagger) произвести тестирование нашего приложения:
    a. Добавить новые сущности (вольер, животное, расписание кормления);
    b. Получить информации о животных, вольерах и расписание кормления;
    c. Выполнить операции кормления, перемещения и т.д.
5. Хранение данных организовать в виде in-memory хранилища (Infrastructure Layer)
6. Написать отчет, в котором отразить:
    a. Какие пункты из требуемого функционала вы реализовали и в каких классах \ модулях их можно увидеть.
    b. какие концепции Domain-Driven Design и принципы Clean Architecture вы применили, скажите в каких классах (модулях).

Критерии оценки

1. Реализация основных требований к функциональности (**2 балла**)
2. Соблюдены принципы Clean Architecture (**4 баллов**):
    a. Слои должны зависеть только внутрь (Domain не зависит ни от чего)
    b. Все зависимости между слоями через интерфейсы
    c. Бизнес-логика полностью изолирована в Domain и Application слоях
3. Соблюдены концепции Domain-Driven Design (**3 баллов**):
    a. Использование Value Objects для примитивов
    b. Инкапсуляция бизнес-правил внутри доменных объектов
4. Покрыто тестами более 65% кода **(1 балл)**

Штрафы

1. до – 2 баллов за наличие ошибки во время выполнения кода;
2. до – 5 баллов, если программа не собирается;
3. – 1 балл за каждый день просрочки дедлайна
4. – 1 балл за грубое игнорирование кодстайла.