Отчет Мини дз 2   
Нгуен Куиет Чыонг БПИ238  
Отчет о проделанной работе

1. Реализация требуемого функционала

a. Добавить/удалить животное

- Контроллер: ZooManagement.Api/Controllers/AnimalsController.cs (методы POST Create и DELETE Delete)

- Репозиторий: ZooManagement.Infrastructure/Repositories/InMemoryAnimalRepository.cs (AddAsync, RemoveAsync)

- Доменная модель: ZooManagement.Domain/Entities/Animal.cs (конструктор, свойства, метод MoveTo)

b. Добавить/удалить вольер

- Контроллер: ZooManagement.Api/Controllers/EnclosuresController.cs (методы POST Create и DELETE Delete)

- Репозиторий: ZooManagement.Infrastructure/Repositories/InMemoryEnclosureRepository.cs (AddAsync, RemoveAsync)

- Доменная модель: ZooManagement.Domain/Entities/Enclosure.cs (конструктор, методы AddAnimal, RemoveAnimal, Clean)

c. Переместить животное между вольерами

- Контроллер: AnimalsController PUT /{id}/move/{enclosureId}

- Сервис: ZooManagement.Application/Services/AnimalTransferService.cs (метод MoveAsync orchestrates удаление из старого и добавление в новый)

- Доменная модель: Animal.MoveTo в ZooManagement.Domain/Entities/Animal.cs; Enclosure.AddAnimal/RemoveAnimal

- Доменное событие: ZooManagement.Domain/Events/AnimalMovedEvent.cs

d. Просмотреть расписание кормления

- Контроллер: ZooManagement.Api/Controllers/FeedingSchedulesController.cs (метод GET)

- Репозиторий: ZooManagement.Infrastructure/Repositories/InMemoryFeedingScheduleRepository.cs

- Доменная модель: ZooManagement.Domain/Entities/FeedingSchedule.cs (свойства Id, AnimalId, FeedingTime, Food, Executed)

e. Добавить новое кормление в расписание

- Контроллер: FeedingSchedulesController POST Create

- Сервис: ZooManagement.Application/Services/FeedingOrganizationService.cs (метод AddFeedingAsync)

- Доменная модель: создание экземпляра FeedingSchedule в Application, метод MarkExecuted для отметки выполнения

- Доменное событие: ZooManagement.Domain/Events/FeedingTimeEvent.cs

f. Просмотреть статистику зоопарка

- Контроллер: ZooManagement.Api/Controllers/StatisticsController.cs (метод GET)

- Сервис: ZooManagement.Application/Services/ZooStatisticsService.cs (метод GetAsync возвращает общее число животных, вольеров и свободных вольеров)

- Используются репозитории IAnimalRepository и IEnclosureRepository

2. Примененные концепции Domain‑Driven Design и принципы Clean Architecture

a. Domain‑Driven Design

- Entities (богатая модель с инкапсулированной бизнес‑логикой): ZooManagement.Domain/Entities/Animal.cs, Enclosure.cs, FeedingSchedule.cs

- Value Objects для примитивов: ZooManagement.Domain/ValueObjects/AnimalName.cs, Species.cs, Food.cs, EnclosureType.cs

- Доменные события: ZooManagement.Domain/Events/AnimalMovedEvent.cs, FeedingTimeEvent.cs

- Инкапсуляция бизнес‑правил внутри доменных объектов: метод Animal.Feed проверяет тип корма, Animal.MoveTo генерирует событие, Enclosure.AddAnimal проверяет вместимость и вызывает Animal.MoveTo, FeedingSchedule.MarkExecuted отмечает выполнение и генерирует событие

b. Clean Architecture

- Разделение на слои:

• Domain (ядро) содержит модели и события и не зависит ни от чего внешнего

• Application (сервисы и интерфейсы) содержит логику бизнес‑операций и интерфейсы репозиториев, зависит только от Domain

• Infrastructure (реализация репозиториев, публикация событий, in‑memory хранилище) зависит от Application.Interfaces и Domain

• Presentation (Web API контроллеры) зависит от Application и Infrastructure, реализует REST‑эндпоинты

- Все зависимости направлены внутрь: контроллеры используют сервисы через интерфейсы; Application зависит только от контрактов; Infrastructure реализует эти контракты

- Внедрение зависимостей через интерфейсы: IAnimalRepository, IEnclosureRepository, IFeedingScheduleRepository, IEventPublisher в ZooManagement.Application/Interfaces

- Настройка DI в ServiceCollectionExtensions.cs в слоях Application и Infrastructure

- Изоляция бизнес‑логики в Domain и Application слоях, контроллеры не содержат логики, только оркестрацию вызовов сервисов и репозиториев

Все перечисленные файлы и модули реализуют заявленные требования и применяют основные принципы DDD и Clean Architecture.  
