Краткое описание проекта

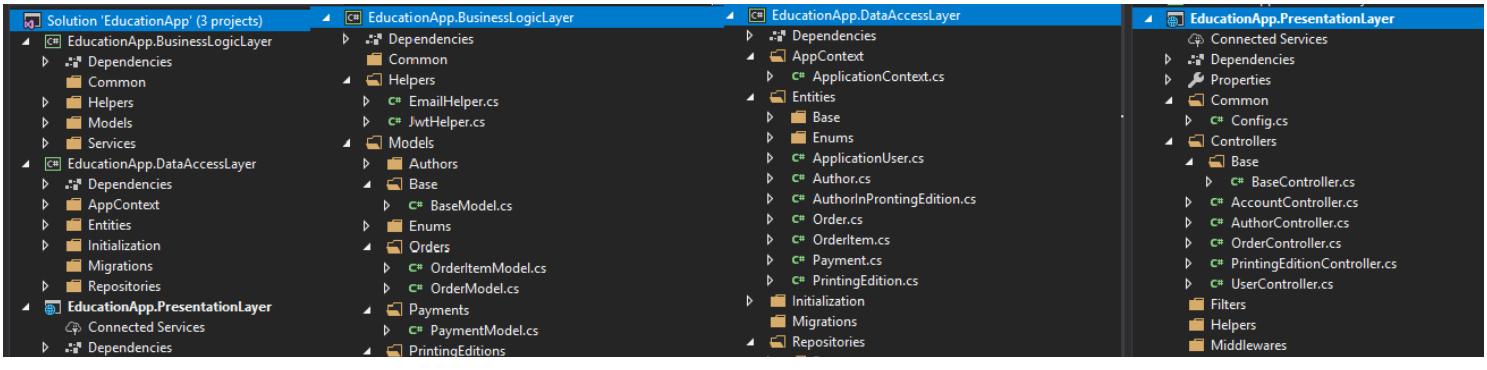
Цель проекта - создание сайта-магазина печатного издания(книги, журналы, газеты), с личным кабинетом для пользователей и администратора. Авторизированный клиент может добавить выбраный товар в корзину с указанием количества и провести оплату. Администратор в свою очередь заполняет данные через админ-панель и просматривает заказы и проведенные платежи (для отправки пользователю товара).

Функционал проекта

1. Роль администратора: авторизация (с подтверждением на email), наполнения контента сайта (добавление печатных изданий, авторов), просмотр заказов и платежей(Orders/order items/payments);
2. Роль пользователя: авторизация (с подтверждением на email), просмотр контента сайта, возможность совершения покупок и оплаты выбранного товара для авторизированных клиентов
3. Контент сайта — книги, журналы и газеты. У каждого печатного издания (ПИ) регламентируется один или более автор. Просматривать товары на сайте могут любые пользователи, однако совершать покупки - только авторизированные клиенты.
4. Поиск ПИ по названию, фильтрация по цене и типу, pagination, добавление выбранного ПИ в корзину. Корзина представлена в виде popup с полем ввода количества заказываемого ПИ, кнопкой оплаты и полем описания заказа; иконка корзины расположена в header.
5. Страница оплаты представлена в виде popup с применением сторонней Api (Stripe integration). Страница результата проведения оплаты представлена в виде popup с сообщением об успешной транзакции и номере заказа, либо с сообщением об ошибке при проведении транзакции.
6. Страница истории покупок пользователя.
7. Страница активных заказов со статусом проведения оплаты всех пользователей для администратора.
8. Дополнительное задание(на усмотрение наставника): CRUD операции для пользователей.

Структура проекта

Структура проекта представляет собой традиционную многослойную архитектуру. Примерная архитектура проекта представлена на рис.1.



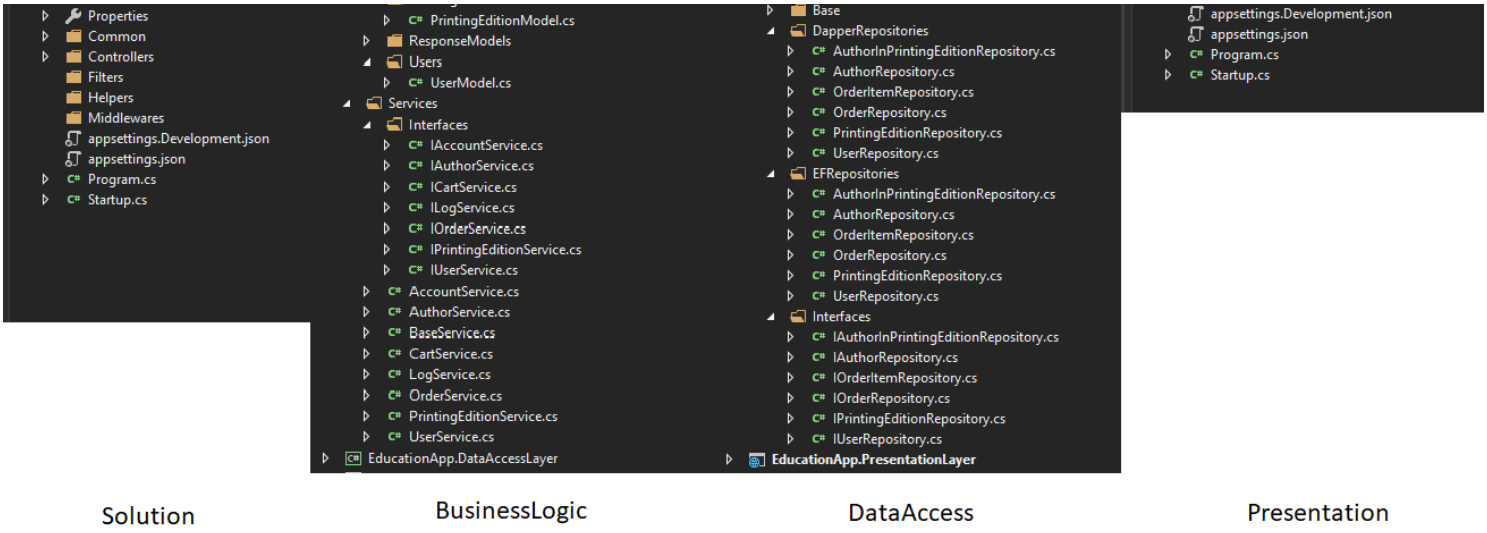


Рис.1 - Структура проекта

В solution входит три основных проекта: Asp.Net Core Web Api и два .Net Core ClassLibrary (слои бизнес логики и работы с данными). База данных для проекта формируется с помошью Entity Framework на осноне сущностей (DAL.Entities). Основная логика приложения сосредоточена в классах сервисов слоя BLL. Взаимодействие с таблицами БД происходит в репозиториях слоя DAL, с помощью которых в проекте реализуется репозиторий-паттерн. Сервисы имеют доступ к репозиториям, а контроллеры - к сервисам. Клиентская часть проекта представлена Angular-приложением, которое вместе с WebApi образуют слой представления. Т.к. слой представления не имеет доступа к слою работы с данными, в слое бизнес логики хранятся модели (BLL.Models), которые необходимы для визуального представления информации из БД. Данные, полученные из репозиториев, преобразуются из сущностей в модели, после чего используются бизнес логикой и представлением для отображения в браузере.

Структура базы данных(примерная)

Структура базы данных представляет собой набор таблиц, каждая из которых хранит данные о конкретных сущностях (Entity). Каждая таблица обязательно имеет поле идентификатора (Id). Ориентировочная схема БД с организацией внешних связей представлена на рис.2.

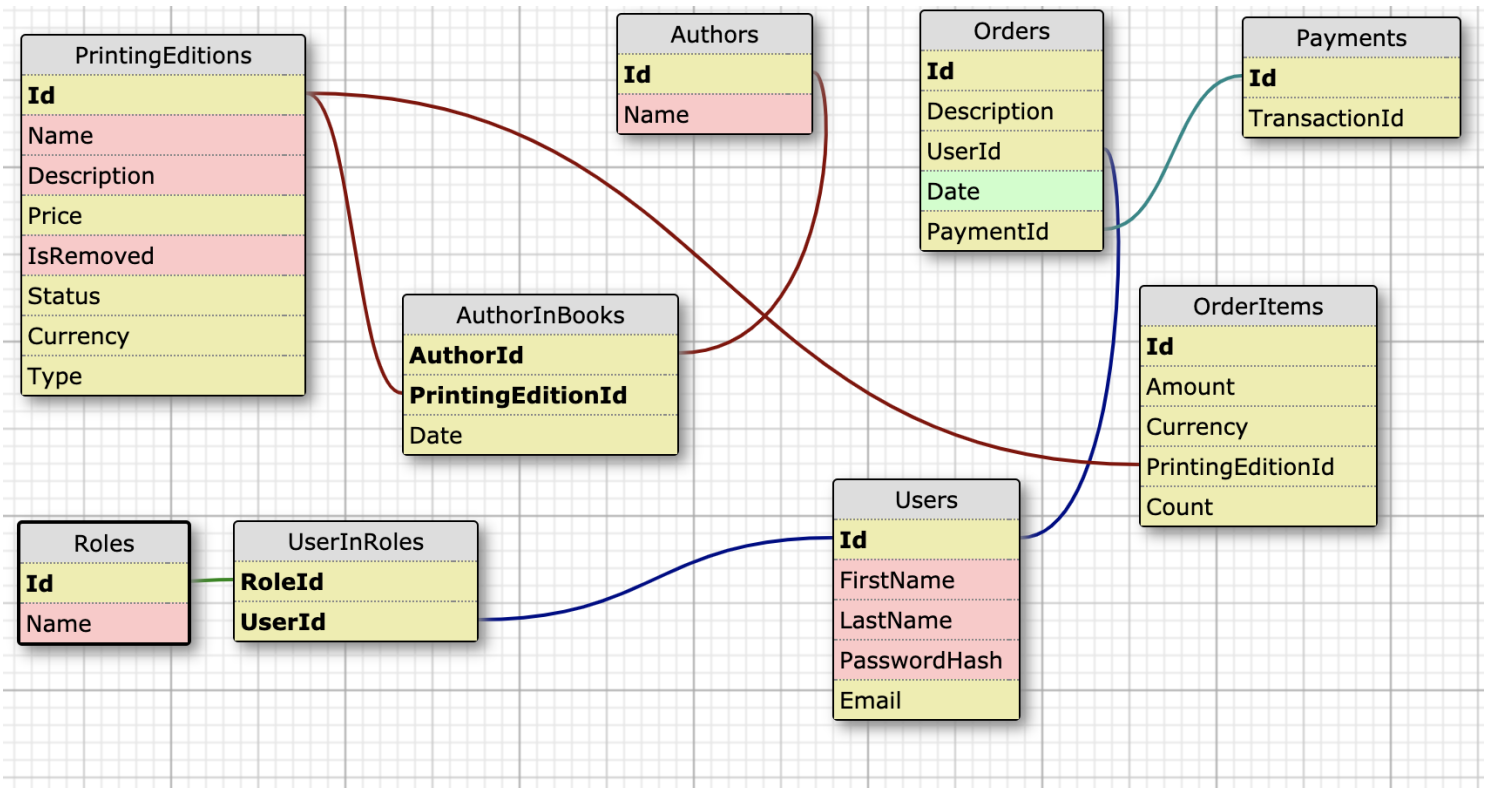


Рис.2 - Ориентировочная схема базы данных

Хранение данных в БД осуществляется согласно следующим принципам:

1. Users, UserInRoles, Roles, добавлены с помощью Identity; необходимо использовать стандартные Identity схемы для пользователей AspNetUsers etc;
2. PrintingEditions содержит в себе все ПИ, к которым в соответствии с типом относятся книги, журналы и газеты; поля Currency и Status являются перечислениями;
3. Базовая сущность проекта должна содержать поля Id, CreationData и IsRemoved (для отметки о том, что объект удален);
4. Таблица AuthorInPrintingEditions (как и AspNetUserRoles) является промежуточной и применяется для связи Author и PrintingEdition по типу «многие ко многим»;
5. Таблицы Orders и OrderItems содержат в себе необходимые данных для оформления заказа; данные из корзины используются для формирования заказа, проведения оплаты (Payments) и связывания их с помощью полей OrderId, UserId и PaymentId.

Последовательность выполнения задания

Выполнение проекта разбито на отдельные этапы, длительность которых — одна неделя. Задание формируется как на неделю, так и на каждый день. Длительность рабочего дня — 8 часов, рабочей недели — 40 часов. Каждую неделю наставник проводит Code review проекта, вносит свои замечания и рекомендации, указывает на ошибки. Все замечания должны быть утсранены до следующего Code review.

Выполнение проекта следует проводить согласно регламенту данного задания. Каждый этап выполнения расписан почасово. Задание на один рабочий день определяются по продолжительности пунктов задания на неделю, исходя из длительности рабочего дня.

Требования к выполнению проекта

В основу проекта должен быть положен принцип многослойности архитектуры с разделением ответственности на слои представления, бизнес логики и работы с данными. Каждый слой должен иметь четко выраженную структуру (см.п. Структура проекта).

Необходимо четко придерживаться правил написания кода, прописаных в файле Code Guide. Именования классов, методов переменных и пр. должны строго соответствовать требованиям. Также должны соблюдаться именования Namespaces проектов и классов с учетом расположения их в дереве каталогов solution.

В процессе добавления контроллеров, сервисов и репозиториев до момента создания соответствующих представлений тестирование ногово функционала следует производить с использованием Postman.

Необходимо использовать перечисления для таких свойств, как User.UserRole(Admin и Client), PrintingEdition.Type(Book, Journal и Newspaper), Order.Status(Paid и Unpaid), PrintingEdition.Currency(USD, EUR, GBP, CHF, JPY, UAH или др.). Не допускается наличие закомментированного кода (весь ненужный код следует удалять).

Design

Макеты представляют приблизительный внешний вид каждой страницы клиентской части проекта. После согласования изменений с наставником допускается изменение дизайна страниц, цветовых схем, стилей, добавление визуальных эффектов.

1 Week

В течении первой недели необходимо выполнить следующие задания:

1. Настройка рабочего ПК (вход в учетную запись VisualStudio, настройка SQL server, установка VSCode, Node.JS, AngularCLI); (2-3 hours)
2. Ознакомление и соблюдение правил Code Guide; (0.5)
3. Ознакомление и соблюдение правил Git Flow; (0.5)
4. Ознакомление с требуемой структурой проекта - см.п.Структура проекта; (0.5)
5. Создание нового solution, внедрение принципа многослойности архитектуры - добавление проектов Presentation (WebApi+Angular), BusinessLogic, DataAccess; (2-3)
6. Создание репозитория BitBucket или Git, добавление проекта в SourceTree, загрузка проекта на репозиторий, создание рабочей ветки, выполнение commit; (1-2)
7. Выполнение настройки проекта: настройка конфигураций development/production, отладка локального IIS, deploy проекта; (6-10)
8. Структурирование проекта (создание необходимых папок, файлов конфигурации и т.д.) - см.п.Требования к выполнению проекта; (2-3)
9. Ознакомление со структурой будущей базы данных (БД), планирование сущностей — см.п.Структура базы данных; (1-2)
10. Создание базовых сущностей, добавление основных сущностей (entities), инициализация миграций; (4-6)
11. Внедрение и настройка механизма DependencyInjection; (1-2)
12. Добавление сервиса логирования ошибок, добавление фильтра ошибок с логированием, проверка корректности работы фильтра; (3-5)

13. Добавление базового сервиса (при необходимости), создание основных сервисов и интерфейсов проекта (слой BLL); (2-4)

2 Week

В течении второй недели необходимо выполнить следующие задания:

1. Создание базового <Generic> репозитория и интерфейса с основным функционалом по работе с БД (EntityFramework + CRUD); (2-4)
2. Интеграция Identity Framework (SignIn, SignUp с подтверждением через Email, восстановление/изменение пароля); (2-4)
3. Добавление user репозитория и интерфейса с применением EntityFramework (получение/создание/проверка роли/добавление роли/аутентификация/изменение/удаление user и др.), используя Identity функционал (таблицы AspNet, UserManager, SignInManager и др.); (4-6)
4. Добавление помощника отправки сообщений на Email с применением SmtpClient; (1-2)
5. Добавление помощника Jwt (генерация access token и refresh token); (2-4)
6. Добавление функционала account сервиса и интерсейса (SignIn на основе Jwt token + RefreshToken, S ignUp с подтверждением аккаунта через Email, восстановление/изменение пароля), использование помощников JWT и Email, добавление моделей; (2-4)
7. Создание базового и основных Api контроллеров (см.п.Структура проекта), внедрение сервисов; (2-3)
8. Добавление функционала account контроллера (SignIn, SignUp, SignOut, ForgotPassword и пр. с использованием Account сервиса); (3-5)
9. Добавление функционала user сервиса и user интерфейса (методы CRUD для user), добавление user модели, добавление функционала user контроллера (администрирование пользователей); (3-5)
10. Добавление функционала user контроллера (profile клиента и администратора, коллекция пользователей для администратора и др.); (3-5)

3 Week

В течении третьей недели необходимо выполнить следующие задания:

1. Создание printingEdition репозитория и интерфейса(EntityFramework + CRUD для printingEdition, коллекция printingEdition по параметрам фильтрации); (2-4)
2. Создание author репозитория и интерфейса(EntityFramework + CRUD для author); (2-3)

Создание author репозитория и интерфейса(EntityFramework + CRUD для author); (2-3)

3. Создание authorInPrintingEditions репозитория и интерфейса(EntityFramework + CRUD для authorInPrintingEditions); (2-3)
4. Добавление функционала author сервиса и интерфейса (CRUD для author), добавление author модели; (2-3)
5. Добавление функционала printingEdition сервиса и интерфейса (CRUD для printingEdition и authorInPrintingEdition), добавление printingEdition модели, добавление методов фильтрации и сортировки; (3-5)
6. Добавление функционала author контроллера (коллекция авторов, выбор авторов, создание и удаление авторов для администратора); (2-3)
7. Добавление функционала printingEdition контроллера (коллекция printingEdition, покупка printingEdition для клиента, создание и удаление printingEdition для администратора); (2-3)
8. Добавление функционала printingEdition сервиса и интерфейса (CRUD для printingEdition и authorInPrintingEdition), добавление printingEdition модели, добавление методов фильтрации; (3-5)
9. Добавление функционала author контроллера (коллекция авторов, выбор авторов, создание и удаление автора для администратора); (2-3)
10. Добавление функционала printingEdition контроллера (коллекция/покупка printingEdition, создание/удаление printingEdition для администратора); (2-3)
11. Интеграция локального хранилища для хранения содержимого корзины покупок клиента (local storage); (1-2)
12. Создание payment репозитория и интерфейса (EntityFramework + CRUD для payment): (1-2)
13. Создание order репозитория и интерфейса (EntityFramework + CRUD для order): (2-3)
14. Создание orderItem репозитория и интерфейса (EntityFramework + CRUD для orderItem): (2-3)

4 Week

В течении четвертой недели необходимо выполнить следующие задания:

1. Добавление функционала cart сервиса, использование local storage(методы CRUD для локальных orderItem), добавление orderItem модели; (2-3)
2. Ознакомление с сервисом Stripe, создание профайла stripe, добавление payment помощника (использование сторонней Api) для проведения транзакции; (4-6)
3. Добавление функционала order сервиса и интерфейса (CRUD для order, payment и orderItem, получение order для клиента и администратора, связывание данных), добавление моделей, использование payment помощника; (6-8)
4. Добавление функционала order контроллера (коллекция order для клиента и администратора, создание order и payment, использование библиотеки Stripe); (3-5)
5. Интеграция Swagger, отображение содержания методов Api контроллеров; (1-2)
6. Применение технологии LazyLoading для загрузки данных; (2-3)
7. Создание базового <Generic> репозитория с основным функционалом по работе с БД (Dapper + CRUD); (2-3)
8. Добавление user репозитория с применением Dapper (в соот-и с функционалом проекта), использование Identity функционала; (3-5)

9. Создание printingEdition репозитория (Dapper + CRUD для printingEdition): (2-4)
10. Создание author репозитория (Dapper + CRUD для author): (2-3)

5 Week

В течении пятой недели необходимо выполнить следующие задания:

1. Создание authorInPrintingEditions репозитория (Dapper + CRUD для authorInPrintingEditions): (2-3)
2. Создание payment репозитория (Dapper + CRUD для payment): (1-2)
3. Создание order репозитория (Dapper + CRUD для order): (2-3)
4. Создание orderItem репозитория (Dapper + CRUD для orderItem): (2-3)
5. Интеграция Swagger, отображение содержания методов Api контроллеров; (1-2)
6. Интеграция представления (SPA – Angular last) проекта — добавление Angular проекта, создание основных модулей (Auth, PrintingEdition, Author, Order, Cart, User, Administrator, Shared), добавление модулей Angular(material, bootstrap, alert, httpClient и др.), структурирование Angular проекта(создание необходимых каталогов: Services, Components, Interfaces, Models, Enums и др.), добавление компонентов shared (Header, Footer и др.), настройка маршрутизации; (4-6)
7. Добавление функционала auth модуля (в соот-и с auth контроллером), интергация необходимых модулей и компонентов (SignIn, SignUp с подтверждением, восстановление и изменение пароля), настройка маршрутизации; (12-16)
8. Создание и добавление функционала auth сервиса (Angular), настройка обработки ошибок; (3-4)

6 Week

В течении шестой недели необходимо выполнить следующие задания:

1. Добавление функционала administrator и user модулей, добавление необходимых компонентов и представлений, добавление user сервиса (Angular); (4-6)
2. Добавление функционала author модуля (в соот-и с author контроллером и userRole), интергация необходимых компонентов, добавление author сервиса (Angular); (5-7);
3. Добавление функционала printingEdition модуля (в соот-и с printingEdition контроллером и userRole), интергация необходимых компонентов, добавление printingEdition сервиса (Angular), применение фильтра коллекции printingEdition, настройка маршрутизации, применение pagination для отображения ПИ; (8-10);
4. Добавление функционала cart модуля (в соот-и с order контроллером), интергация необходимых компонентов, добавление cart сервиса (Angular), интеграция модуля Stripe; (4-6);
5.

Добавление функционала order модуля (в соот-и с order контроллером и userRole), интергация необходимых компонентов, добавление order сервиса (Angular), применение pagination; (6-8);

6. Проверка правильности выполнения задания, итоговый Code rewiew (проверка соответствия code style), исправление ошибок и замечаний; (5-10).