Лабораторная работа №1

В ходе выполнения лабораторной работы вам будет необходимо выполнить следующие действия:

1. Установить Visual Studio 2022.
2. Установить .NET версии 6.0.
3. Создать проект по шаблону – web api
4. Запустить проект и продемонстрировать открывшуюся страницу в браузере
5. Инициализировать директорию с проектом для системы контроля версий и добавить ваш проект в созданный репозиторий.
6. Установить библиотеку для взаимодействия с базой данных.
7. Реализовать класс Product, который инкапсулирует товар в интернет-магазине.
8. Разработать интерфейс IProductService, определяющий операции с товаром (добавление, удаление, изменение, поиск).
9. Разработать класс ProductService, реализующий интерфейс IProductService
10. Создать контроллер ProductController, принимающий пользовательские запросы и возвращающий результат выполнения операции в формате json

Ход выполнения лабораторной работы

Пункт 1 – 2. Установка Visual Studio 2022 и .Net

Среда разработки доступна для скачивания с официального сайта [Microsoft](https://visualstudio.microsoft.com/ru/). Далее вам необходимо установить среду выполнения .Net версии 6.0. Это можно сделать с помощью Visual Studio Installer либо скачать с официального сайта (runtime и sdk).

Далее в консоли (powershell) выполните команду:

*dotnet --list-sdks*

*dotnet --list-runtimes*

Списки не должны быть пусты. **Скриншот с выводом команд показать преподавателю.**

Случае если вы используете Linux. Тогда вы можете выбрать один из следующий вариантов:

* Установить Visual Studio Code и .Net с помощью менеджера пакетов из официальных репозиториев. Инструкции по установки для каждого дистрибутива операционной системы Linux доступны на сайте Microsoft или сайте самого производителя дистрибутива.
* Установить Rider от JetBrains. Возникнет сложность с поисков ключа для активации среды разработки(~~GitHub и Google~~).

Пункт 5 - Начало.

Создайте учётную запись на Gitlab. Далее создайте на сайте Gitlab репозиторий, в котором будет храниться проект. Во время создания обязательно создайте файл .gitignore – выберите шаблон для .Net (в случае отсутствия найдите шаблон в открытых репозиториях на Gitlab/GitHub и добавьте необходимый текст в файл .gitignore). После этого выполняем следующую команду:

git clone <url>, url – ссылка на ваш проект на GitHub/GitLab

Пункт 3. *Сначала сделайте Пункт 5 – Начало*.

Данный пункт предполагает, что вы успешно выполнили предыдущие пункты. Из-за того, что вы можете использовать разные средства разработки, будет предложено универсальное средство создания проекта нашего приложения.

Запустите терминал (PowerShell в Windows и Terminal в Linux) в каталоге, в котором находится файл .gitignore скопированного проекта. Подряд выполняем следующие команды:

1. dotnet new sln --name Magazine

Создаём файл решения. Файл решение – текстовое описание проектов, из которых состоит «общий проект».

1. dotnet new classlib --name Magazine.Core

Создаём проект по шаблону «библиотека классов», в котором будут располагаться основные модели (классы), отвечающие за логику работы приложения.

1. dotnet new webapi --name Magazine.WebApi

Создаём проект по шаблону «webapi», который будет являться серверной частью приложения, то есть обрабатывать пользовательские запросы.

1. dotnet sln .\Magazine.sln add .\Magazine.Core\Magazine.Core.csproj

Добавили ссылку на проект библиотеки классов.

1. Аналогично добавьте в решение ссылку на проект webapi.
2. dotnet add .\Magazine.WebApi\Magazine.WebApi.csproj reference .\Magazine.Core\Magazine.Core.csproj

Ссылка на проект означает, что все сущности доступные в библиотеке классов будут доступны в проекте webapi. Проект библиотеки классов – отдельная библиотека (.dll), с которой можно взаимодействовать. На практике вы всегда используете какие-то библиотеки для решения задач. Они могут быть самописными или скачаны из репозиториев. Всё зависит от решаемых задач.

После этого откройте файл решения .sln в среде разработки. Вы увидите, что решение состоит из 2 проектов. Авто запускаемым проектом является проект webapi. При этом из-за того, что проект webapi ссылается на библиотеку классов, сначала будет скомпилирована библиотека классов (файл .dll), и только затем будет скомпилирован проект webapi.

Пункт 5 – Конец.

Теперь, когда у нас есть проект магазина, вам необходимо перенести изменения с локальной машины на сервер. Для этого выполните следующие команды (работу продолжаем в том же каталоге):

1. git add \*
2. git commit -m ‘comment’
3. git pull
4. git push

В терминале появится сообщение о том, что изменения отправлены на сервер. Покажите преподавателю добавленный проект на сервер.

Пункт 4

Покажите преподавателю запущенный проект webapi.

Ответьте на следующие вопросы:

1. Почему в браузере не открылась главная страница сайта?
2. На каком порту запущен сайт?

Пункт 6.

В качестве базы данных будем использовать SQLite. Использование этой БД связано с тем, что она требует минимальное количество ресурсов, не требует установки сервера базы данных в отличии от таких баз данных, как Postgres, MySql, SqlServer и прочих.

Установку необходимой библиотеки будем производить через терминал (да, среды разработки поддерживают графический интерфейс, с помощью которого возможно установить, но в данном случае необходимо наработать навыки работы с терминалом). Для установки библиотеки выполним следующую команду в терминале:

cd Magazine.WebApi

dotnet add package Microsoft.EntityFrameworkCore.Sqlite

dotnet tool install --global dotnet-ef

dotnet add package Microsoft.EntityFrameworkCore.Design

Покажите преподавателю список добавленных пакетов(библиотек) с помощью консоли: dotnet list package

Пункт 7.

В проекте Magazine.Core создайте папку Models, в ней создайте класс Product со следующими полями:

* Id – уникальный идентификатор (рассмотрите [Guid](https://learn.microsoft.com/ru-ru/dotnet/api/system.guid?view=net-8.0))
* Definition – Текстовое описание товара
* Name – Название товара
* Price – стоимость товара
* Image – изображение товара

Тип данных выбирайте самостоятельно.

Пункт 8.

В проекте Magazine.Core создайте папку Services. В ней создайте интерфейс IProductService, который описывает следующий функционал:

* Метод Add – добавляет в базу данных новый элемент. Возвращает сущность, которая была создана во время добавления. В случае возникновения исключения производить обработку в вызывающем методе.
* Метод Remove – удаляет из базы данных соответствующий элемент. Возвращает сущность, которая была удалена из базы данных. В случае возникновения исключения производить обработку в вызывающем методе.
* Метод Edit – изменяет соответствующий элемент. Возвращает сущность, которая была изменена в базе данных. В случае возникновения исключения производить обработку в вызывающем методе.
* Метод Search – поиск элемента по переданным параметрам. Возвращает сущность, которая была найдена в базе данных. В случае отсутствия сущности возвращать null. В случае возникновения исключения производить обработку в вызывающем методе.

Пункт 9. В проекте Magazine.WebApi добавить класс ProductService реализующий интерфейс IProductService.

Пункт 10. Если вы создали проект правильно и установили нужную версию среды выполнения, то у вас в проекте Magazine.WebApi должен автоматически быть создан класс – WeatherForecastController. По аналогии с данным контроллером, вам необходимо самостоятельно разработать класс ProductController с 4 методами, с помощью которых можно будет использовать функционал класс ProductService.

***При выполнении пунктов лабораторной работы необходимо изменения программного кода добавлять на сервер системы контроля версий.***