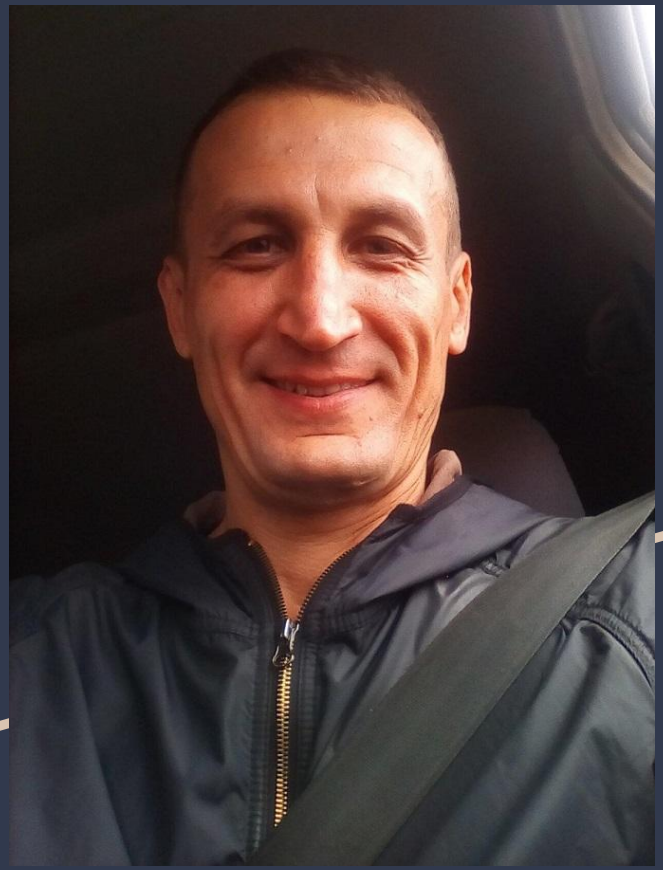
Сервис для замеров артериального давления, сохранения и визуализаций вносимых данных и комментариями к ним.

Веб-приложение для среднего и пожилого возраста, для ежедневного внесения данных, анализа и мониторинга.

Антонов Сергей



О себе:

- Родился и проживаю в городе Уфа.
- Интересы – спорт, шахматы. В юности учился в музыкальной школе , инструмент баян, играю на гитаре.
- Опыт работы: производство, предпринимательство, продажи.
- Сайт-портфолио: http://xn--d1aqu.su/duma

Поставленная задача

В рамках проекта будет разработан веб-сервис на C# с применением шаблона паттерна MVC. Проект будет предоставлять API для создания, редактирования, удаления и получения информации о вносимых данных и их визуализацию в таблицах и виде графиков.

Создание и подключение к базе данных Mysql, с помощью Entity Framework Core, которое представляет ORM-решение.

Реализация систему двухфакторной аутентификации.

Разработка дизайна и структуры сайта.

Клиентская часть будет реализовывать код в виде javascript.js с сочетанием vue .js библиотеки.

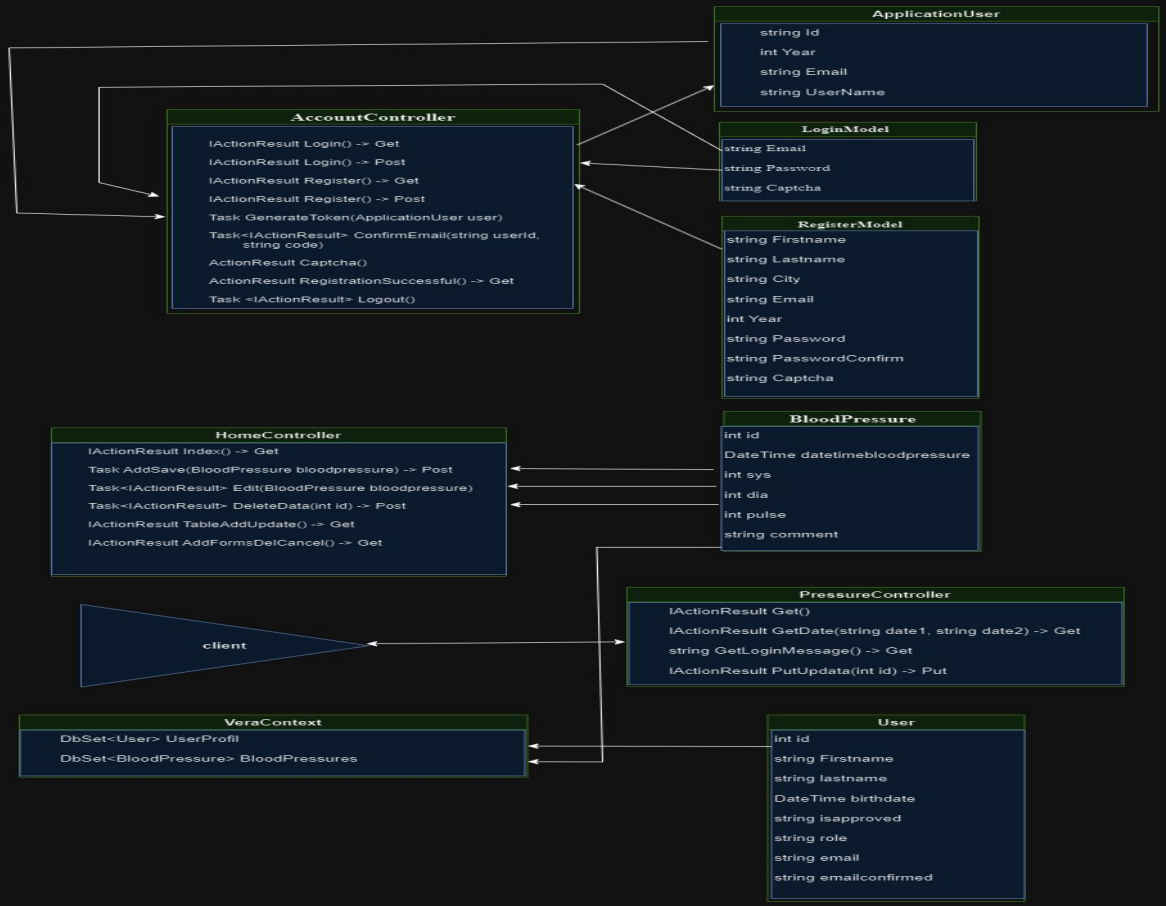
План проекта

Решение задачи проходит с использованием среды Microsoft Visual Studio, язык программирования C#.

Структура создания приложения:

- Разработка структурной схемы приложения Net Core на сервере , создание моделей, контроллеров, представлений и конфигураций приложения.
- Структура клиентской части - форма входа и регистраций пользователя, страница профиля, добаление, удаление, обнавление данных. Используемые библиотеки Javascript / Vue.js / jquery / html / css / bootstrap, страница входа с аутентификацией и валидацией.
- Создание структуры базы данных – MySQL.
- Развертывание проекта на хостинге .

Структура приложения на сервере



Описание структуры на сервере

Здесь мы видим 3 контроллера, которые из них представляют функционал для регистраций и входа в сервис , пользовательскую страницу где разворачивается основной функционал для сервиса и встроенный Api - контроллер, который возвращает json-данные. Далее наблюдаем присутствие моделей для работы с бизнес-логикой. И в итоге структуры имеется объект контекста для взаимодействия с базой данных в виде сущностей.

Структура на клиенте

start_page

- async function GetRegister()
- async function LoginIn()
- function OrientationMobil(img_captcha)

js_add_measuring

- function IsEmail(email)
- function ValidateEmail(email)
- ValidateCountry(country)
- async function RowsAddUpdateTable()
- async function AddFormDelCancel()

modul_table

- async function GetLoginMessage()
- function IconEventClickUpdata(event)
- function IconEventClickDelete(event)
- async function DeleteData(id, numberRow)
- async function PutUpData(id, index)
- async function FormLoading(modelBloodPressures, index)
- function ChartShow()
- async function GetDate()

Validate

- function validRegister(event)
- function validInput(event)
- async function ValidateAddSave(event, model)

Описание структуры на клиенте

Структура на клиенте имеет в себе реализацию стартовой страницы, в которой также имеете функционал для регистраций пользователя и входа в сервис.

Добавление динамического пользовательского интерфейса, который заносит данные в таблицу замеров пользователя.

Валидация введенных данных на клиенте и их отображения для информирования пользователя.

Структура базы данных

Описание структуры базы данных

Структура базы данных имеет связи в виде “один ко многим” сущности users к добавляемых пользователем данных, а также сущности aspnetusers которые позволяет регистрироваться, логинится, аутентифицироваться в сервисе с помощью двухфакторной реализаций JWT-токена.

Пользовательский интерфейс

Описание структуры интерфейса

Пользовательский интерфейс представляет из себя простой интуитивно понятную навигацию, который имеет выборку по датам замеров пользователя, добавление замеров в таблицу пользователя, редактирование, удаление, а также прокрутка скролла. Ниже видим их визуализацию в виде графика, который имеет зум, для приятного удобного пользования.

Достигнутые цели

В данном проекта, заложены и применены современный подход к разработке веб-приложения, в которых объекты классов имеют слабосвязанные связи в виде интерфейсов, что предполагает в будущем, для расширения функционала. Структура взаимодействия приложения с базой данных также является расширяемой для интеграций новых связей. Подключение новых сервисов в конвейер запросов и ответов, также имеет интуитивный простой и быстрый подход для развертывание нового сервиса. Аутентификация пользователя имеет современную двухфакторную аутентификацию благодаря JWT- токенов, что в современном мире сегодня крайне важно. Авторизация также имеет простой подход который применен в виде атрибутов к конечным точкам.

В проекте применены шаблоны в виде паттернов, такие как MVC, где модели, представления и бизнес-логика структурирована, которые легко совмещены с Api REST, что выгодно отличается для быстрого получения необходимых данных в виде Json – строк. Также продолжая с темой запросов и ответов, видим что имеют асинхронную работу, что конечно же добавляет приятною для пользователя и функционала в целом, удобной и практичной. На клиенте применены и использованы ряд библиотек, такие как Vue.js, что конечно же выгодно отличается от старых сборок , которые не отличались грамотным использованием памяти в стеке и куче. И конечно же хочется сказать о графике вывода данных, которые просто внедряются и подгружают необходимые данные и их визуализируют.

- проект развернут на хосте по адресу: http://xn--d1aqu.su:8081/vera/

Идеи на будущее

Идеи будущего видеться, в улучшений и добавления нового функционала. Здесь необходимо будет добавить новый функционал с подключением искусственного интеллекта, разделение функционала на микро-сервисы и развертыванием в докер-контейнере. Добавить проверку на ошибки(exception) ввода пользователя. Реализовать добавление и удаление пользователя в операционной системе, для аутентификация пользователя на хосте. Добавления тестов на конечные точки.