МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к дипломному проекту на тему:**

«Фитнес-социальная сеть "SportClub"»

Дипломник Першай Ян Борисович

(Ф.И.О.)

Руководитель проекта ст. преподаватель Блинова Е.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В. В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты: к.т.н., доц. Жиляк Н. А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

ст. препод. Евлаш А. И

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер асс., Ковалевская Н. И.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Дипломный проект защищен с оценкой

Председатель ГЭК к.т.н., доц. Дюбков В. К.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Минск 2021

**Содержание**

реферат;

содержание;

введение;

1: постановка задач и обзор аналогов;

* 1. Аналитический обзор литературы
  2. Проблемы мотивации при ведении ЗОЖ
  3. Достоинства фитнес-социальной сети
  4. Постановка задачи
  5. Обзор среды разработки Visual Studio
  6. Обзор среды разработки Visual Studio Code
  7. Обзор платформы .NET Core
  8. Обзор языка C#
  9. Обзор мобильного фреймворка Flutter
  10. Обзор СУБД MSSQL Server
  11. Обзор облачного сервиса Azure
  12. Обзор аналогичных программных решений
      1. Веб-приложение Up4Sport
      2. Веб-приложение Facepoint
      3. Мобильное приложение Fitior
  13. Вывод по разделу

2: проектирование программного средства;

2.1 Основные технические требования к разработке

2.2 Разработка архитектуры проекта

2.3 Функциональные возможности проекта

2.4 Структурная схема мобильного приложения

2.5 Диаграмма вариантов использования

2.6 Проектирование Базы данных

2.7 Блок-схема создания поста

2.8 Вывод по разделу

3: разработка программного средства;

* 1. Технологии, используемые в серверной части приложения
  2. Реализация серверной части приложения
     1. Взаимодействие с базой данных
     2. Маппинг запросов
     3. Dependency injection
     4. Обработчики запросов
     5. Создание контроллеров
     6. Реализация аутентификации и авторизации
     7. Выгрузка фото пользователя в Azure Blob Storage

3.3 Технологии, используемые в клиентской части приложения

3.3.1 Реализация клиентской части приложения

3.3.2 Взаимодействие с серверной частью приложения

3.4 Вывод по разделу

4: анализ информационной безопасности приложения;

4.1 Реализация аутентификации и авторизации с использованием JWT-токена

4.2 Хеширование паролей в базе данных

4.3 Вывод по разделу

раздел 5: тестирование программного средства;

5.1 Тестирование серверной части приложения

5.2 Тестирование клиентской части приложения

5.3 Вывод по разделу

6: руководство пользователя;

6.1 Системные требования оборудования для запуска сервера

6.2 Необходимое программное обеспечение

6.3 Системные требования устройства для пользования мобильным приложением

6.4 Руководство по использованию мобильного приложения

6.5 Регистрация пользователя

6.6 Авторизация пользователя

6.7 Поиск и подписка на пользователей

6.8 Оценивание публикации

6.9 Добавление новой публикации

6.10 Добавление тренерской публикации

6.11 Сохранение публикации и просмотр сохраненных публикаций

6.12 Редактирование аккаунта и профиля

6.13 Выход из аккаунта

6.14 Удаление аккаунта

6.15 Вывод по разделу

7: технико-экономическое обоснование проекта

7.1 Общая характеристика разрабатываемого программного средства

7.2 Исходные данные и маркетинговый анализ

7.3 Методика обоснования цены

7.3.1 Объем программного средства

7.3.2 Основная заработная плата

7.3.3 Дополнительная заработная плата

7.3.4 Отчисления в Фонд социальной защиты населения

7.3.5 Расходы на материалы

7.3.6 Расходы на оплату машинного времени

7.3.7 Прочие прямые затраты

7.3.8 Накладные расходы

7.3.9 Сумма расходов на разработку программного средства

7.3.10 Расходы на сопровождение и адаптацию

7.3.11 Полная себестоимость

7.3.12 Определение цены, оценка эффективности

7.4 Выводы по разделу

заключение;

список использованных источников;

приложения и графическая часть

**Введение**

На сегодняшний день важно вести здоровый образ жизни. Физические нагрузки и проведение регулярных тренировок позволяют привести себя в форму, держать тело в тонусе, увеличить энергию тела, предотвратить множественные заболевания, улучшить сон, моральное состояние и многое другое. Преимущества физических нагрузок можно перечислять долго.

Для проведения регулярных тренировок важно соблюдать режим, держать дисциплину, тренировать силу воли чтобы не пропускать тренировки.

На сегодняшний день большое распространение получают фитнес-мобильные приложения. Они позволяют проводить тренировки самостоятельно, изучить тематические рекомендации от авторов приложения, напомнить пользователю о необходимости проведения тренировки по расписанию. Иногда, приложения могут заменить тренера при проведении стандартных тренировок.

Но многие популярные фитнес приложения имеют функционал, обеспечивающий набор тренировок от авторов приложения, которым необходимо следовать и не дают возможность поделиться опытом со своей стороны, показать свои успехи.

Целью моего дипломного проекта является разработка фитнес-социальной сети SportHub, предоставляющей доступ к статьям на тематику спорта и позволяющую делиться опытом путём публикации собственных статей на тему здорового образа жизни, спортивного питания и др., осуществлять взаимную подписку. В тексте публикаций можно ставить новые цели и отмечать свои успехи, благодаря чему появляется ответственность перед теми, кто наблюдает за успехами пользователя и становится легче соблюдать дисциплину и режим тренировок. В публикациях можно задать интересующий вопрос и получить ответ в комментариях от более опытных пользователей. В разделе статей имеется набор тематических статей, написанных опытными авторами и, в случае, если пользователь хочет делиться своим опытом на большую аудиторию, у него будет возможность попытаться попасть в команду авторов.

Для достижения цели сформулированы следующие задачи:

* Анализ предметной области
* Проектирование программного средства
* Разработка программного средства
* Обеспечение информационной безопасности программного средства
* Тестирование программного средства
* Создание руководства пользователя

1. **Постановка задачи и обзор аналогов**

Перед началом работы необходимо составить набор задач для решения их при разработке программного средства и провести обзор аналогов для обнаружения достоинств и недостатков в данной сфере.

**1.1 Проблемы мотивации при ведении здорового образа жизни**

Очень часто, приняв решение вести Здоровый образ жизни, что подразумевает в себе пробежки, тренировки, правильное питание, человеку необходимо соблюдать этот режим, что подразумевает в себе регулярность, постоянство выполнения физических упражнений. И если на первых порах регулярно тренироваться, следить за питанием хватает мотивации и силы воли, то со временем она угасает и человеку становится сложнее себя контролировать.

Несмотря на то, что большинство понимает, что тренировки полезны для здоровья, но возможность встретиться с друзьями, отдохнуть с семьёй, усталость после работы все же перевешивают. Опираясь на мотив «должен сделать потому что полезно» требует борьбы со своей ленью, проявление силы воли, что большинство не любит и старается избегать.

Ответственность только лишь перед собой и своей силой воли часто приводит к окончанию режима тренировок и ведению здорового образа жизни.

В этом случае очень важно себя мотивировать. Ещё важнее мотивировать себя правильно. Необходимо ставить себе правильные и конкретные цели. Отмечать успехи, вести отчет. Планировать. Таким образом, гораздо эффективнее ставить себе мотивации, цели, имеющие ключевые слова «стать», «быть».

Большую роль играет ответственность не только перед собой, но и ещё перед кем-либо. Можно договориться с другом контролировать друг друга или регулярно публиковать свой прогресс в социальных сетях. В этом случае гораздо сложнее будет бросить тренировки, показав тем самым слабую силу воли и проигрыш перед единомышленниками, которые наблюдали прогресс.

* 1. **Достоинства фитнес-социальной сети**

Развитие интернета, а в частности социальных сетей, привело к тому, что они проникли практически в каждую сферу нашей жизни. В различных тематических социальных сетях можно найти работу, общаться со знакомыми, выбрать и приобрести автомобиль, обсуждать книги, музыку, информационные технологии, бизнес, следить за спортивными событиями и обсуждать их с единомышленниками.

Очень удобно, когда для каждой сферы можно найти соответствующую социальную сеть. Таким образом, соц. сети, не имеющие определенной тематики, не будут отображать специализированный контент, предназначенный для узкой аудитории, а пользователям легче найти единомышленника.

Ниже приведены основные преимущества и недостатки создания и существования специализированной фитнес-социальной сети.

Преимущества:

* наличие конкретной тематики;
* построение своего комьюнити с течением времени;
* повышенная эффективность рекламы;
* возможность легко найти единомышленников;
* лёгкий поиск нужного контента и сообществ.

Недостатки:

* сложность привлечения пользователей из аналогичных, более развитых, не тематических соц. сетей;
* повышенная конкуренция за рекламодателя;
* отклонения от тематики соц. сети не очень уместны.

**1.3 Постановка задачи**

Главной целью дипломного проекта является разработка клиент-серверного мобильного приложения, которое обеспечит пользователям доступ к тематической спортивной социальной сети, предоставляющей доступ к статьям на тематику спорта и позволяющую делиться опытом путём публикации собственных статей на тему здорового образа жизни, спортивного питания и др., осуществлять взаимную подписку.

Большую перспективу в нынешнее время имеют именно тематические социальные сети, объединяющие в себе сообщества единомышленников.

**1.4 Обзор среды Visual studio**

Написание оптимизированного и правильного кода невозможно без надежной и удобной среды разработки. Хорошая среда разработки помогает повысить эффективность работы программиста обеспечивая мгновенную детализацию кода и помогая избегать некоторых наиболее распространенных синтаксических ошибок. Эффективные редакторы кода способны уведомить программиста об ошибках в синтаксисе непосредственно при написании кода. Многие среды разработки имеют возможность дополнительных улучшений с помощью полезных расширений. Сам процесс написания кода становится более увлекательным и комфортным для разработчика.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

При разработке дипломного проекта использовалась среда разработки Visual Studio 2019. На рисунке 1.1 показана скриншот, на котором отображена среда разработки с открытым проектом и несколькими окнами основных инструментов, которые используются при разработке программного продукта:

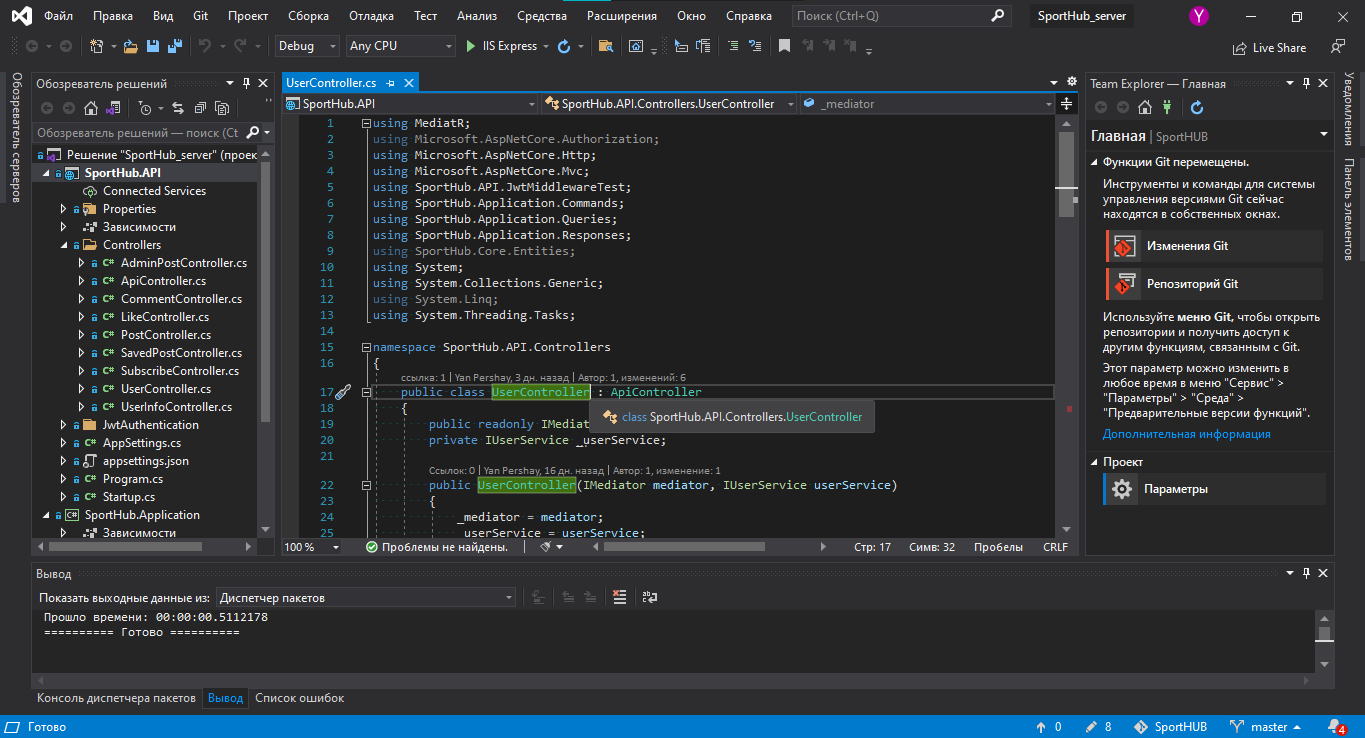


Рисунок 1.1 – скриншот среды разработки Visual Studio 2019

Основными инструментами при разработке являются:

* обозреватель решений (слева) позволяет просматривать файлы кода, перемещаться по ним и управлять ими. Обозреватель решений позволяет упорядочить код путем объединения файлов в решения и проекты;
* в окне редактора (центр) отображается содержимое файла. Здесь можно редактировать код или разрабатывать пользовательский интерфейс, например окно с кнопками или текстовые поля;
* Team Explorer (справа) позволяет отслеживать рабочие элементы и использовать код совместно с другими пользователями с помощью технологий управления версиями, таких как Git и система управления версиями Team Foundation (TFVC).

Рассмотрим несколько дополнительных и полезных функций IDE Visual Studio 2019:

* создание облачных приложений для Azure. Visual Studio предлагает набор инструментов, позволяющих с легкостью создавать облачные приложения на базе Microsoft Azure. Она упрощает настройку, сборку, отладку, упаковку и развертывание приложений и служб в Microsoft Azure прямо из IDE. Чтобы получить инструменты Azure и шаблоны проектов, при установке Visual Studio выберите рабочую нагрузку Разработка для Azure.
* подключение к базам данных. Обозреватель сервера позволяет просматривать экземпляры и ресурсы SQL Server в локальной или удаленной среде, в Azure, Salesforce.com, Microsoft 365 и на веб-сайтах, а также управлять ими. Чтобы открыть обозреватель серверов, выберите в главном меню Вид > Обозреватель серверов.
* отладка, тестирование и совершенствование кода. При написании кода требуется запустить его и проверить на ошибки и производительность. Современная система отладки Visual Studio позволяет выполнять отладку кода в локальном проекте, на удаленном устройстве или в эмуляторе устройства. Можно просматривать код с шагом в один оператор, проверяя значения переменных. Можно задать точки останова, которые срабатывают только при выполнении указанного условия. Параметры отладки можно контролировать в самом редакторе кода, не покидая окно с кодом.

**1.5 Обзор редактора кода Visual Studio Code**

Написание оптимизированного и правильного кода невозможно без надежного и удобного редактора кода. Хороший текстовый редактор помогает повысить эффективность работы программиста обеспечивая мгновенную детализацию кода и помогая избегать некоторых наиболее распространенных синтаксических ошибок. Эффективные редакторы кода способны уведомить программиста об ошибках в синтаксисе непосредственно при написании кода. Многие текстовые редакторы имеют возможность дополнительных улучшений с помощью полезных расширений. Сам процесс написания кода становится более увлекательным и комфортным для разработчика.

Visual Studio Codeявляется бесплатным кроссплатформенным оптимизированным редактором кода. Программа имеет открытый исходный код. В основе редактора лежит Electron – фреймворк для создания «нативных» десктопных приложений с помощью веб-технологий (HTML, CSS, JS) и Node.js. Он оснащен доступным набором инструментов для редактирования, отладки, выполнения задач и управление версиями, а также обладает поддержкой более 30 языков программирования и форматов файлов. Редактор легко интегрируется с другими сервисами как Unity3D, Visual Studio Online, GitHub.

Как и многие другие редакторы кода, VS Code принимает общий пользовательский интерфейс и макет проводника слева, показывая все файлы и папки, к которым у вас есть доступ, и редактор справа, показывая содержимое файлов, которые вы открыли. Пользовательский интерфейс разделен на пять областей:

* редактор. Основная область для редактирования ваших файлов. Можно открыть столько редакторов, сколько захотите, закрепив их вертикально или горизонтально;
* боковая панель. Содержит различные виды, такие как Explorer, чтобы помочь вам при работе над проектом;
* строка состояния. Информация об открытом проекте и файлах, которые вы редактируете;
* панель активности. Расположена в крайнем левом углу, позволяет переключаться между представлениями и дает дополнительные индикаторы, зависящие от контекста, например, количество исходящих изменений при включении Git;
* панель. Редактор может отображать различные панели под областью редактора для вывода или отладки информации, ошибок и предупреждений или встроенного терминала. Панель также может быть перемещена вправо для большего вертикального пространства.

Пользовательский интерфейс редактора удобный и интуитивно понятный при желании его можно видоизменять в настройках или с помощью расширений. На рисунке 1.2 показана среда Visual Studio Code с открытым проектом.

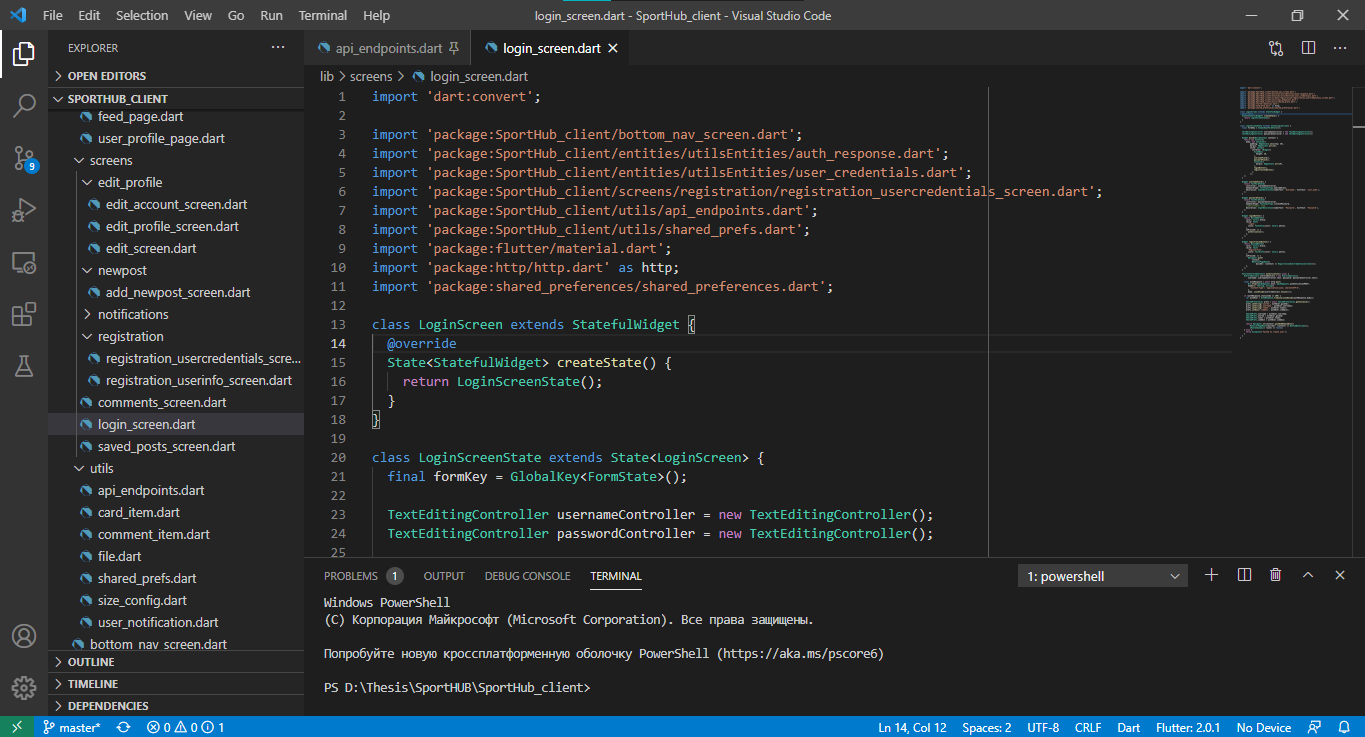


Рисунок 1.2 – Интерфейс Visual Studio Codeс открытым проектом

Каждый раз при запуске VSCode, он открывается в том же состоянии, в котором он был в последний раз. Папка, макет и открытые файлы сохраняются. Открытые файлы в каждом редакторе отображаются с заголовками вкладок в верхней части области редактора;

Рассмотрим несколько полезных и интересных функций Visual Studio Code:

* intelliSense. Это технология автодополнения и подсказок, которая дописывает название функций и переменных при вводе начальных букв;
* code Anilizer. Функционал, помогающий найти ошибки в коде;
* updates Manage*r*. Менеджер плагинов, позволяющий находить, устанавливать и обновлять пакеты или дополнения;
* views. Вкладки для отображения различной информации;
* zen-mode. Режим позволяет сосредоточиться на своем кода, скрывая весь пользовательский интерфейс, кроме редактора (без панели активности, строки состояния, боковой панели и панели), переходя на полный экран и центрируя макет редактора;
* minimap. Мини карта, являющаяся схемой кода, которая дает общий обзор исходного кода, который полезен для быстрой навигации и понимания кода.

Для Visual Studio Code существует большое количество расширений, которые можно скачать и установить прямо в редакторе.

При разработке клиентской части дипломного проекта я использовал Visual Studio Code. Загрузить и установить его можно на официальном сайте Microsoft, продукт является абсолютно бесплатным и легковесным для компьютера. Выбор был сделан в его пользу из-за быстроты разработки клиентской части дипломного проекта и возможности работы с фреймворком Flutter и языком Dart, а также большому количеству удобных расширений.

**1.6 Обзор платформы .NET Core**

.NET Core — новейшая универсальная платформа разработки, поддерживаемая Microsoft. Она работает на разных платформах и была переработана таким образом, чтобы сделать .NET быстрым, гибким и современным. Это является одним из основных вкладов Microsoft. Теперь есть возможность создавать приложения для Android, iOS, Linux, Mac и Windows с помощью .NET.

Ниже приведены основные характеристики .NET Core:

* открытый исходный код. .NET Core — это реализация с открытым исходным кодом, использующая лицензии MIT и Apache 2.
* кроссплатформенность. Приложение, реализованное в .NET Core, может быть запущено, и его код может быть повторно использован независимо от платформы. В настоящее время поддерживается три основные операционные системы (ОС): Windows, Linux, MacOS.
* модульность. .NET Core выпускается через NuGet в небольших сборочных пакетах, что позволяет разработчикам оптимизировать приложение, включая только те пакеты, которые необходимы в проекте.

**1.7 Обзор языка программирования C#**

C# (произносится как "си шарп") — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать множество типобезопасных и надежных приложений, работающих в экосистеме .NET.

C# относится к широко известному семейству языков C, и покажется хорошо знакомым любому, кто работал с C, C++, Java или JavaScript.

Основные функции языка C#, которые позволяют создавать надежные и устойчивые приложения:

* сборка мусора – автоматически освобождает память, занятую недоступными неиспользуемыми объектами;
* типы, допускающие значение null – обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты;
* обработка исключений – предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них;
* лямбда-выражения – поддерживают приемы функционального программирования;
* синтаксис LINQ – создает общий шаблон для работы с данными из любого источника и возможность создания удобной выборки данных.
* поддержка асинхронности – предоставляет синтаксис для создания распределенных систем.

В C# действует единая система типов. Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями ссылочные типы, так и типы значений. C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность.

В C# особое внимание уделяется управлению версиями для обеспечения совместимости программ и библиотек при их изменении. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Программы C# выполняются в .NET, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (CLR) и набор библиотек классов. Среда CLR — это реализация общеязыковой инфраструктуры языка (CLI), являющейся международным стандартом, от корпорации Майкрософт. CLI является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке C# компилируется в промежуточный язык (IL), который соответствует спецификациям CLI. Код на языке IL и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением .dll. Сборка содержит манифест с информацией о типах, версии, языке и региональных параметрах для этой сборки.

При выполнении программы C# сборка загружается в среду CLR. Среда CLR выполняет JIT-компиляцию из кода на языке IL в инструкции машинного языка. Среда CLR также выполняет другие операции, например, автоматическую сборку мусора, обработку исключений и управление ресурсами. Код, выполняемый средой CLR, иногда называют "управляемым кодом", чтобы подчеркнуть отличия этого подхода от "неуправляемого кода", который сразу компилируется в машинный язык для определенной платформы.

Обеспечение взаимодействия между языками является ключевой особенностью .NET. Код IL, созданный компилятором C#, соответствует спецификации общих типов (CTS). Код IL, созданный из кода на C#, может взаимодействовать с кодом, созданным из версий .NET для языков F# , Visual Basic, C++ и любых других из более чем 20 языков, совместимых с CTS. Одна сборка может содержать несколько модулей, написанных на разных языках .NET, и все типы могут ссылаться друг на друга, как если бы они были написаны на одном языке.

В дополнение к службам времени выполнения .NET также включает расширенные библиотеки. Эти библиотеки поддерживают множество различных рабочих нагрузок. Они упорядочены по пространствам имен, которые предоставляют разные полезные возможности: от операций файлового ввода и вывода до управления строками и синтаксического анализа XML, от платформ веб-приложений до элементов управления Windows Forms. Обычно приложение C# активно используют библиотеку классов .NET для решения типовых задач.

**1.8 Обзор мобильного фреймворка Flutter**

Flutter — комплект средств разработки и фреймворк с открытым исходным кодом для создания мобильных приложений под Android и iOS, а также веб-приложений с использованием языка программирования Dart, разработанный и развиваемый корпорацией Google.

Первая версия выпущена в 2015 году под названием «Sky», работала только для Android-приложений. Основная заявленная особенность — высокая графическая производительность (возможность рендеринга 120 фреймов в секунду). Полная поддержка создания веб-приложений появилась в версии 2.0 (март 2021 года), также разрабатывается поддержка создания настольных приложений для Windows, macOS и Linux и Google Fuchsia (в виртуальной машине Dart с JIT-компилятором).

Из-за ограничений на динамическое выполнение кода в App Store, под iOS Flutter использует AOT-компиляцию. Широко используется такая возможность платформы Dart, как «горячая перезагрузка», когда изменение исходного кода применяется сразу в работающем приложении без необходимости его перезапуска.

Основные составляющие комплекта — платформа Dart, движок Flutter, библиотека Foundation, наборы виджетов и средства разработки (Flutter DevTools).

Движок Flutter написан преимущественно на C++, поддерживает низкоуровневый рендеринг с помощью графической библиотеки Google Skia, имеет возможность взаимодействовать с платформозависимыми SDK под Android и iOS.

Библиотека Foundation, написанная на языке Dart, содержит основные классы и методы для создания приложений Flutter и взаимодействия с движком Flutter.

Дизайн пользовательского интерфейса приложений Flutter предполагает использование виджетов, описываемых как неизменяемые объекты какой-либо части пользовательского интерфейса. Все графические объекты, включая текст, формы и анимацию, создаются с помощью виджетов; комбинированием простых виджетов создаются сложные виджеты. С фреймворком поставляется два основных набора виджетов — Material Design (стиль Google) и Cupertino (стиль Apple). При этом создавать приложения Flutter можно и без виджетов, напрямую вызывая методы библиотеки Foundation для работы с канвой.

**1.9 Обзор СУБД MSSQL Server**

Microsoft SQL Server — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

**1.10 Обзор облачного сервиса Azure**

Облачные службы Azure являются примером концепции платформа как услуга (PaaS). Так же, как и служба приложений Azure, эта технология предназначена для поддержки масштабируемых, надежных и недорогих в эксплуатации приложений. Облачные службы Azure размещаются так же, как и службы приложений на виртуальных машинах (ВМ). Однако над виртуальными машинами доступен больший контроль. На виртуальных машинах, использующих облачные службы Azure, можно установить собственное программное обеспечение, а затем получить удаленный доступ к нему.

Дополнительный контроль, помимо прочего, усложняет использование. Если вам не требуются возможности дополнительного контроля, как правило, гораздо быстрее и проще запустить и выполнить веб-приложение в веб-приложениях службы приложений, чем в облачных службах Azure.

Существует два типа ролей облачных служб Azure:

* Веб-роль: автоматически развертывает и размещает приложения с помощью IIS.
* Рабочая роль: не использует IIS и запускает приложение автономно.

Ключевыми особенностями облачных сервисов Azure являются:

* масштабирование и управление. При использовании облачных служб Azure пользователь не создает виртуальные машины. Вместо этого предоставляется файл конфигурации, сообщающий Azure, сколько ресурсов ему требуется, например три экземпляра веб-роли и два экземпляра рабочей роли. Платформа создаст их самостоятельно. Пользователю нужно выбрать размер для резервных виртуальных машин, но не требуется явно создавать их самостоятельно. Если приложение сталкивается с высокой нагрузкой, есть возможность запросить дополнительные виртуальные машины, после чего Azure создаст необходимые экземпляры. В случае спада нагрузки можно завершить работу этих экземпляров и перестать оплачивать их.
* мониторинг. Кроме того, облачные службы Azure обеспечивают мониторинг. Как и модель виртуальных машин, эта модель обнаруживает неисправный физический сервер и перезапускает выполняемые на нем виртуальные машины на новом компьютере. Однако модель облачных служб Azure также обнаруживает не только аппаратные сбои, но и неисправные виртуальные машины и приложения. В отличие от виртуальных машин, она использует агент внутри каждой веб-роли и рабочей роли, что позволяет в случае сбоя запустить новые виртуальные машины и экземпляры приложений.

**1.11 Обзор аналогичных программных решений**

В настоящее время представлено не так много социальных сетей на спортивную тематику, особенно в форме мобильных приложений. Преимущественно – это новостные спортивный сайты, веб-приложения с наличием возможности вести собственный профиль и осуществлять коммуникацию.

**1.11.1 Веб-приложение Up4Sport**

Одной из известных социальных сетей на спортивную тематику является Up4sport. Она представлена в форме веб-приложения (сайта).

Сайт представляет большой функционал по погружению в спортивную тематику. После регистрации пользователь получает возможность создать свой личный спортивный блог, наблюдать за спортивными блогами других пользователей, найти для себя тренера или же партнёра среди пользователей сети, присоединиться в тематические группы к единомышленникам, следить за новостями и отслеживать спортивные события в своем городе.

Скриншот главной страницы сайта представлен на рисунке 1.3:

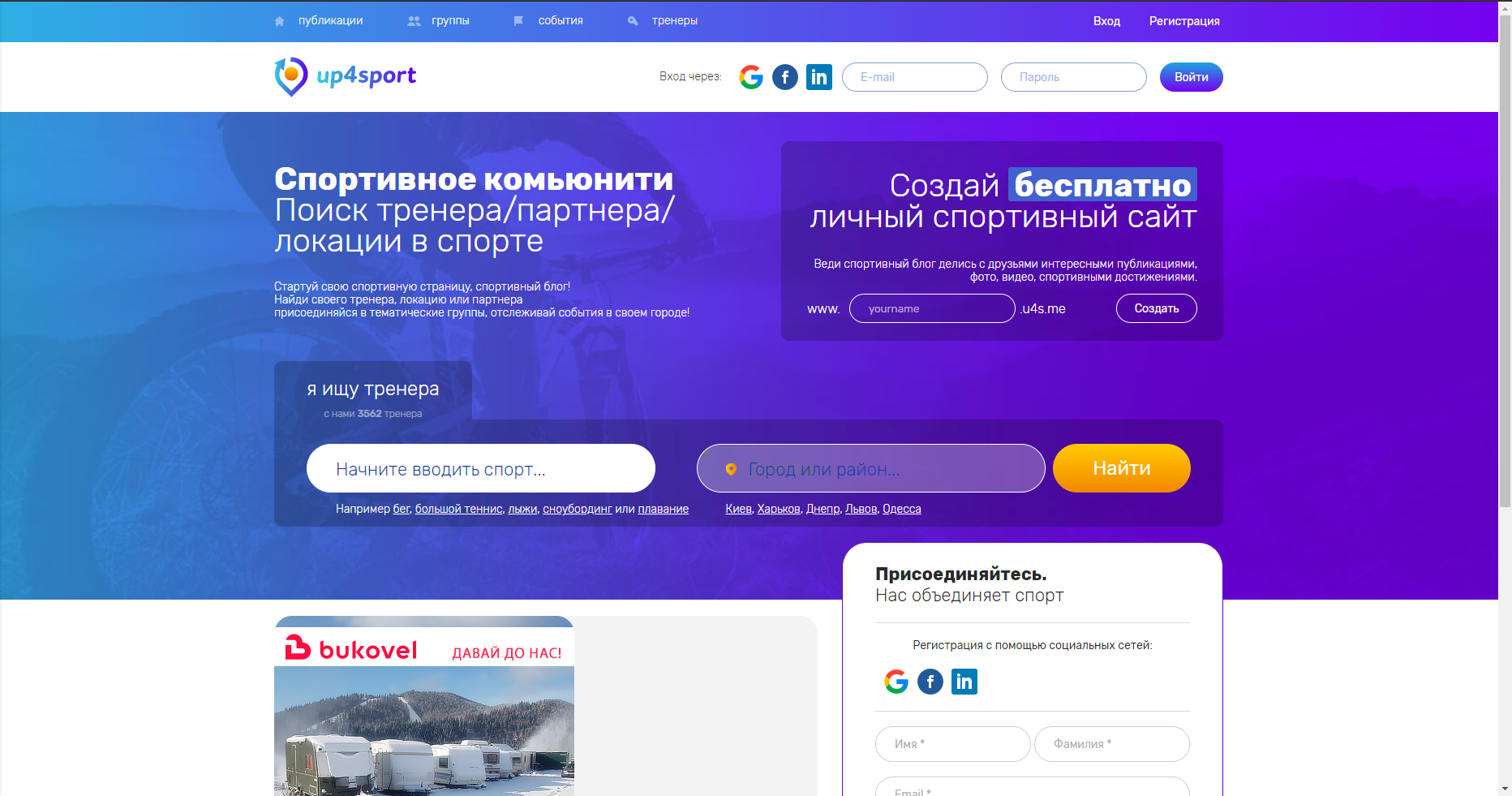


Рисунок 1.3 – Главная страница up4sport

Плюсы социальной сети up4sport.com:

* возможность создания блога с уникальным веб-адресом;
* возможность наблюдать за конкретным видом спорта;
* возможность найти тренера или партнера;
* возможность наблюдать за новостными событиями своего города;

Минусы:

* отсутствие мобильной версии социальной сети;
* ведение блога подразумевает в себе соблюдения определенных правил публикуемого контента;

Up4sport отличный представитель социальных сетей на спортивную тематику, но большим минусом является именно отсутствие мобильной версии приложения.

**1.11.2 Веб-приложение FacePoint**

Следующей социальной сетью на спортивную тематику является FacePoint – путеводитель по спорту. Она представлена в форме веб-приложения (сайта).

Сайт является одной из первых созданных социальных сетей в России, Украине и Беларуси, направленной на объединение всех любителей спорта. У пользователей есть возможность объединяться на одном проекте и общаться на актуальные темы, просматривать блоги, новости, видео, фотогалерею, общаться на форуме и др.

Скриншот главной страницы сайта представлен на рисунке 1.4:

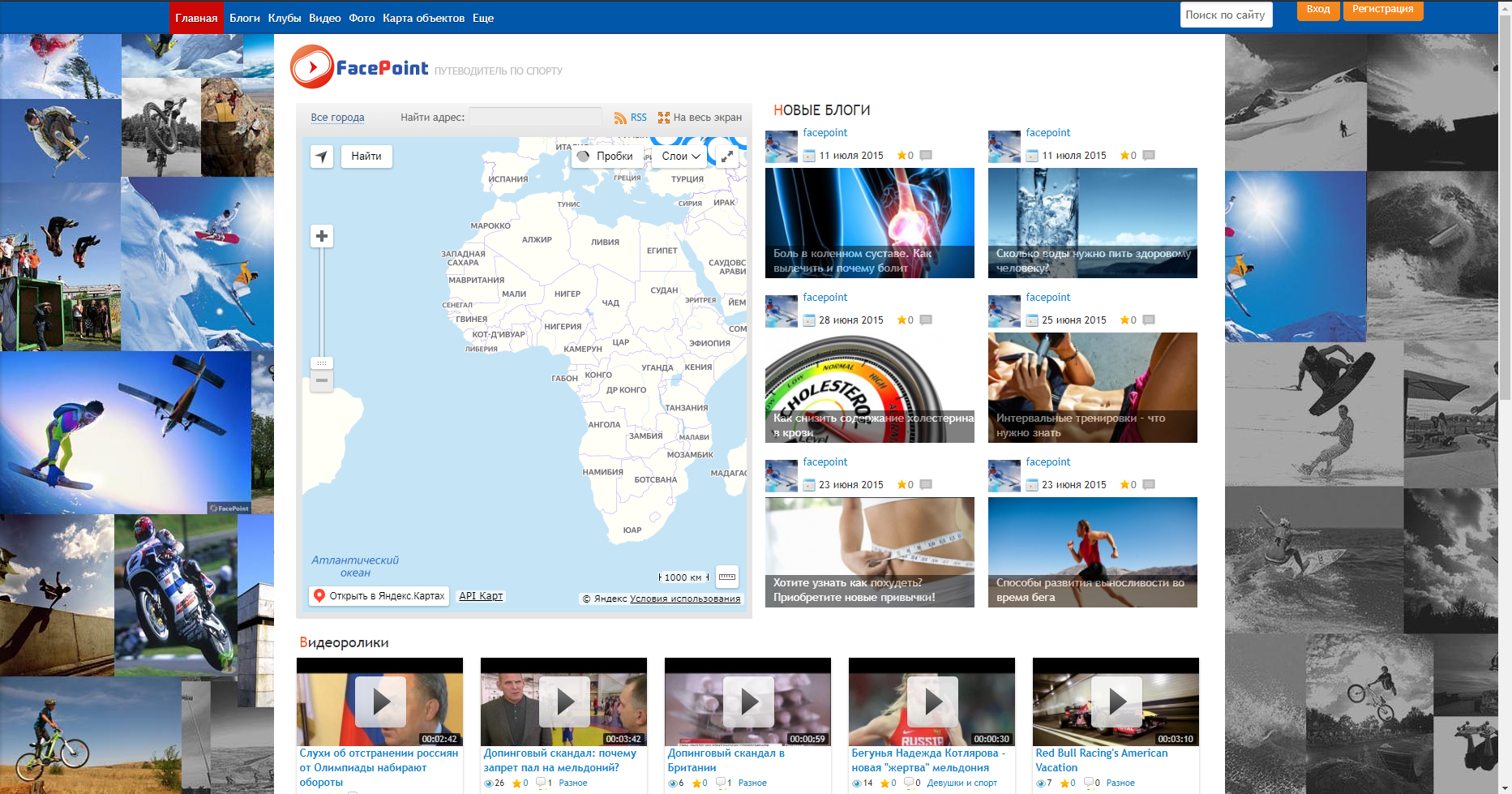


Рисунок 1.4 – Главная страница FacePoint

Плюсы социальной сети facepoint.ru:

* возможность наблюдать за конкретным видом спорта;
* возможность создать и вести собственный блог;
* большое количество новостей и статей;
* возможность наблюдать за тематическими блогами;

Минусы:

* сайт не активен последние несколько лет;
* имеет устаревший дизайн;
* регистрация осуществляется только по приглашениям;
* отсутствие мобильной версии социальной сети;

FacePoint – отличный пример спортивной социальной сети, но главной ее проблемой является то, что сайт утратил популярность, практически не осталось активных пользователей, а следовательно остановилось и развитие проекта.

**1.11.3 Веб-приложение Fitior**

Мобильное приложение Fitior является отличным представителем современной и многофункциональной спортивной социальной сети.

Приложение Fitior предназначено для тех, кто бегает, плавает, ездит на велосипеде, ходит в тренажерный зал — и хочет это делать веселее. С ним можно найти единомышленников в спорте или посмотреть, где и каким спортом занимаются ваши друзья сейчас, рассказать о своих успехах, вести фотожурнал тренировок, открыть для себя новые места и новые спортивные занятия. Сейчас в Fitior более 45 видов спорта, включая одиночные, групповые и игровые.

Скриншот приложения представлен на рисунке 1.5:

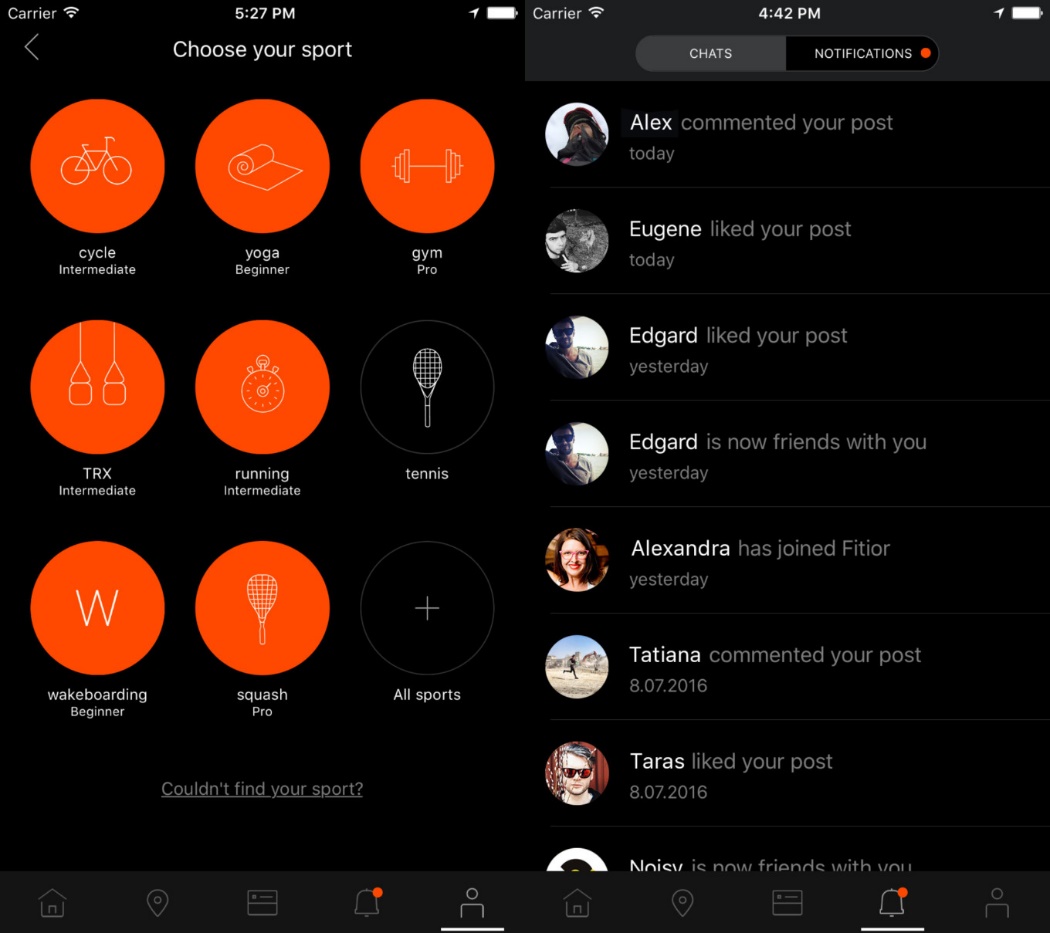


Рисунок 1.5 – приложение Fitior

Плюсы мобильного приложения Fitior:

* возможность поиска единомышленников по виду спорта и по дистанции;
* возможность создания собственного профиля с фото и выбором интересующих видов спорта;
* возможность ведения чата;
* возможность ведения ленты активности;

Минусы:

* отсутствие поддержки на ОС Android;

Fitior – современная и перспективная спортивная социальная сеть, но большим минусом является именно отсутствие поддержки на распространенную ОС Android.

**1.12 Вывод по разделу**

В ходе написания первой главы были сформулированы цели и задачи для разработки дипломного проекта. Сформулирована основная проблематика, натолкнувшая на создание проекта с тематикой спортивной социальной сети. Был подобран и описан основной стек технологий, который будет использоваться при разработке клиентской и серверной части проекта. Сделан обзор на существующие аналоги, где были выделены их преимущества и недостатки, которые необходимо учитывать при разработке программного средства.

Подводя итоги, можно сказать, что информации, накопленной в данном разделе, достаточно для начала проектирования и разработки программного средства.

1. **Проектирование программного средства**

Главной задачей дипломного проекта является создание клиент-серверного мобильного приложения, предоставляющего функционал спортивной социальной сети. Целью проектирования является определение свойств системы на основе сформулированных требований.

При проектировании приложения требуется учесть множество факторов, влияющих на работоспособность создаваемого приложения. Ключевыми критериями являются: удобство и понятность интерфейса, гибкость использования интерфейса, быстрота и эффективность работы приложения и простота в использовании. Для корректного проектирования системы требуется поэтапная реализация проекта.

Этапы проектирования системы:

* предварительные исследования;
* формулировка концепции проекта;
* разработка технического задания;
* проектирование архитектуры серверной части проекта;
* разработка дизайна клиентской части проекта;
* программная реализация системы;
* интеграция клиентской и серверной части;
* устранение недостатков;
* тестирование приложения.

**2.1 Основные технические требования к разработке**

Мобильное приложение обеспечивает функционал спортивной социальной сети. Были поставлены следующие цели для разработки и реализации:

* обладать удобным интерфейсом;
* обладать несколькими пользовательскими ролями;
* возможность создания и редактирования профиля пользователя;
* авторизация пользователя;
* возможность создания собственных публикаций с выбранными изображениями;
* возможность осуществления подписки на пользователей;
* возможность просмотра статей тренера и создания таковых при наличии роли тренера;
* возможность просмотра личного профиля и профилей других пользователей;
* возможность оценивания публикации лайком и комментарием;
* возможность добавления публикации в избранное.

Была произведена оценка требований, обзор наиболее подходящих для реализации проекта технологий. Серверная часть проекта реализовывается на платформе .NET Core. Клиентская часть реализовывается с помощью мобильного фреймворка Flutter. Для хранения данных используется СУБД MS SQL Server 2019. Изображения будут храниться в Azure Storage Blob.

* 1. **Разработка архитектуры проекта**

От выбора архитектуры сервера зависит простота разработки приложения и эффективность работы. Программу с хорошей архитектурой проще расширять, тестировать, отлаживать, а код становится более понятным и читабельным. Для сервера была выбран вид архитектуры .NET приложений «Чистая архитектура (Clean architecture)».

* 1. **Функциональные возможности проекта**

Для моделирования функционала системы важно составить диаграмму вариантов использования, описывающую функциональность и поведение приложения, позволяющее заказчику, разработчику или конечному пользователю лучше понимать сущность приложения.

В результате была разработана диаграмма вариантов использования, на которой изображен функционал разрабатываемого программного средства. С её использованием будет происходить дальнейшая разработка приложения. Диаграмма вариантов использования представлена на рисунке 2.1 и в приложении А:

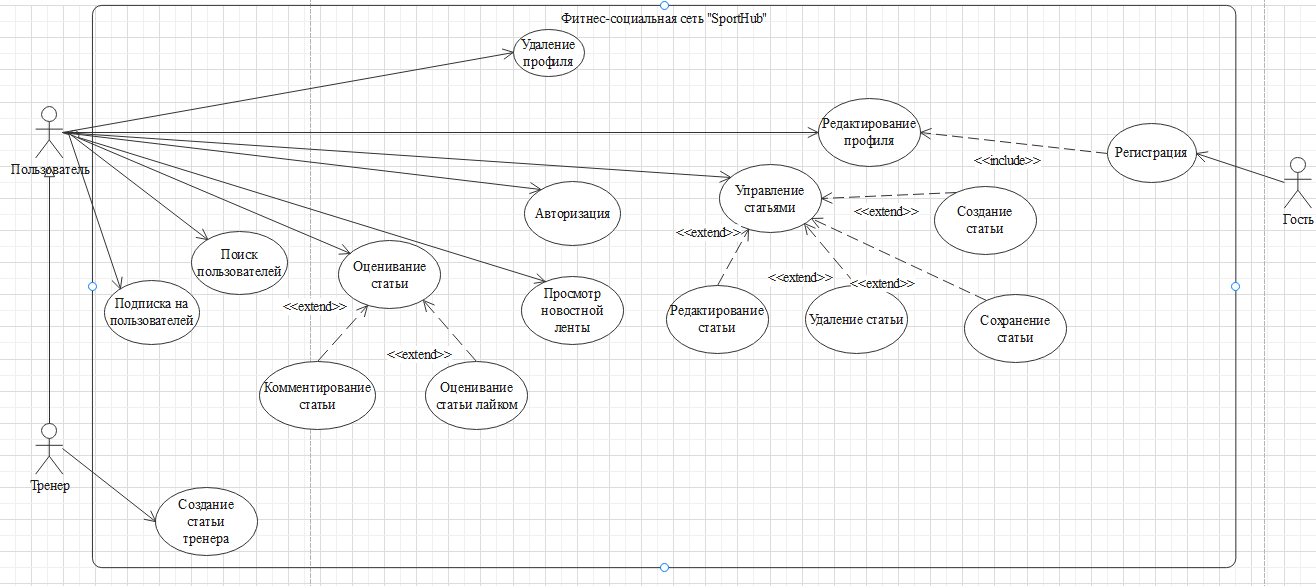


Рисунок 2.1 – Диаграмма вариантов использования