Разработанные программы, тесты и полученные результаты доступны по ссылкам:
1) https://github.com/astraszab/TSP-algorithms;
2) https://github.com/dmitriyklebanov/tsp.

Список использованных источников:
1. Алгоритмы. Построение и анализ / Т. Кормен [и др.] ; пер. с англ. — 3-е изд. — М. : Вильямс, 2015. — 1328 с.
2. Скиена, С. Алгоритмы. Руководство по разработке / С. Скиена ; пер. с англ. — 2-е изд. — СПб. : БХВ, 2017. — 760 с.

МОБИЛЬНОЕ ПРИЛОЖЕНИЯ ДЛЯ ИНДИВИДУАЛЬНЫХ ЗАНЯТИЙ СПОРТОМ

Бастун А.Н.

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Таборовец В.В. – к.т.н.,доцент

Рассматриваются вопросы создания мобильного приложения для фиксации и автоматического анализа результатов контроля данных о физическом состоянии здоровья человека с использованием платформы iOS.

В современном мире вопрос укрепления здоровья человека становиться все более изученным и количество факторов влияющих на него становится все больше. У людей становится все больше носимых устройств, способных собирать данные о состоянии человека в течении дня, таких как телефоны, умные браслеты, часы и т.д. С помощью этих данных можно строить планы тренировок, рекомендации по питанию и т.д. Поскольку стоимость работы персонального тренера, диетолога или другого специалиста является существенной для большинства, либо у людей просто нет на это времени. Использование простого приложения для тренировок и отслеживания текущих показателей может быть большим плюсом в укреплении здоровья, развития общего физического состояния и т.д

Цель разработки является создание мобильного приложения для платформы iOS, способного считывать данные о физическом состоянии пользователя и его активности и предлагать рекомендации в зависимости от целей пользователя. Данные будут считываться в зависимости от имеющихся устройств (телефон, часы и т.д.) и в зависимости от данных разрешений на использование пользователем. Различные международные исследование показали, что постоянная умеренная физическая нагрузка благотворно влияет на работу мозга и сердечно-сосудистой системы, сокращает риски возникновения различных заболеваний [2].

Основной целью приложения является возможность предоставить пользователю более легкий способ следить за своим физическим состоянием и получать максимальную пользу из физических нагрузок тратя минимальное количество времени. Для считывания данных пользователя будет использоваться нативный фреймворк HealthKit, который предоставляет из себя библиотеку различных показателей физического состояния и активностей [1]. Данный фреймворк находится в экосистеме iOS и дает возможность интеграции с различными носимыми устройствами, также содержит различную медицинскую информацию пользователя. Активно используется различными медицинскими учреждениями в Америке для постоянного наблюдения за пациентами[1, 3].

Данные, которые будет использовать приложение:
1) частота пульса;
2) количество шагов сделанных пользователем;
3) количество энергии использованной в состоянии покоя;
4) количество энергии потраченной на активную деятельность;
5) количество времени проведенное в движении (легкой прогулки или более интенсивно);
6) количество потребленной энергии из еды;
7) Опросы пользователей для получение различных данных, которые нет возможности получить с носимых устройств.

Анализ собранных данных производится на основе анализа интенсивности нагрузки (частота пульса и скорость восстановления пульса до нормальных значений), объема нагрузки (количество сделанных шагов, количество времени и энергии потраченное на активную деятельность) и различные опросы и тесты, сделанные пользователем на основе опросников.

Приложение реализует возможность просмотра предлагаемых рекомендаций, выбор и установку целей, личных параметров и других дополнительных параметров, просмотр статистики.

Список использованных источников:

1. HealthKit Depelover guidance [Электронный ресурс]. - Электронные данные. Режим доступа: https://developer.apple.com/documentation/healthkit
2. Harvard Health Publishing [Электронный ресурс]. - Электронные данные. Режим доступа: https://www.health.harvard.edu
3. Human Interface Guidelines [Электронный ресурс]. - Электронные данные. Режим доступа: https://developer.apple.com/design/human-interface-guidelines/

БАЛАНСИРОВКА НАГРУЗКИ МЕЖДУ УЗЛАМИ ПРИ ПОСТРОЕНИИ РАСПРЕДЕЛЕННЫХ СИСТЕМ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вабищевич Е. Г..

Таборовец В.В. – к.т.н.,доцент

Работа посвящена вопросам исследования существующих способов построения распределенных систем с целью создания базовой архитектуры распределенной системы обработки запросов.

Распределенная система — это набор независимых компьютеров, представляющийся их пользователям единой объединенной системой [1]. Среди эмпирических свойств распределенной системы можно выделить следующие:

1) устойчивость системы при выходе из строя одного или нескольких элементов;
2) возможность изменения конфигурации системы в процессе работы;
3) отсутствие полных знаний о системе в одном конкретном элементе системы.

В процессе исследования распределенных систем были рассмотрены следующие архитектуры: архитектура клиент-сервер, архитектура peer to peer, кластерная архитектура.

Кластеры распределения нагрузки – способ организации программного обеспечения и системного оборудования, при котором происходит распределение запросов через один или несколько входных узлов. После этого из входных узлов запросы перенаправляются на обработку в вычислительные узлы. Данный подход обладает следующими преимуществами:

- возможность горизонтального масштабирования;
- равномерное распределение нагрузки между узлами;
- обеспечение высокой доступности системы. Данный пункт достигается благодаря возможностям первых двух пунктов.

В результате рассмотрения принципов построения распределенных систем была разработана следующая базовая архитектура.

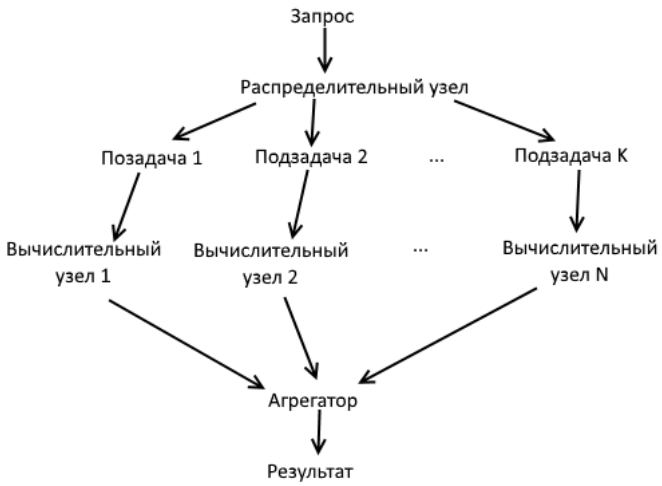

Рис. 1 - Базовая архитектура распределенной системы

Описание рисунка 1: в систему приходит запрос → запрос обрабатывается распределительным узлом систем, разбивается на подзадачи меньшего масштаба → каждая из подзадач отправляется на один из вычислительных узлов → вычислительный узел выполняет обработку подзадачи; результат выполнения отправляется агрегатору → агрегатор дожидается обработки всех подзадач запрос, собирает из них итоговый результат, отдает результат клиенту.