2. Unmanned Aerial Vehicle/ Wikipedia // [Электронный ресурс] – 2021. – URL: https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle. – Дата доступа: 20.03.2021.

3. Ордоди Мартон. Дельтапланеризм/ История авиации и воздухоплавания // [Электронный ресурс] – 1984. – URL: http://fly-history.ru/books/item/f00/s00/z0000012/index.shtml. – Дата доступа: 20.03.2021.

4. Richard Vaughn. The difference between Cartesian, Six-Axis and SCARA robots/ MachineDesign // [Электронный ресурс] – 2013. – URL: https://www.machinedesign.com/mechanical-motion-systems/article/21831692/the-difference-between-cartesian-sixaxis-and-scara-robots/. – Дата доступа: 02.12.2013.

5. NanoPi Neo Core2 / Wikipedia // [Электронный ресурс] – 2019. – URL: https://wiki.friendlyarm.com/wiki/index.php/NanoPi_NEO_Core2/. – Дата доступа: 07.11.2019.

В. В. Расторгуев
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **М. А. Подалов,** ст. преподаватель

РАЗРАБОТКА WEB-СЕРВЕРА СИСТЕМЫ «УМНЫЙ ДОМ» НА ANGULAR 11

С каждым днём человек все больше и больше погружается в мир, который раньше показывали по телевизору с подписью «Фантастика». Мы привыкли к высокому уровню жизни и продолжаем повышать планку. Дом – это то место, в котором люди проводят большую часть времени. Из-за этого возникла потребность быстрее справляться с рутинными домашними делами. Важным открытием в этой сфере стала разработка системы «Умный Дом», которая не только решала поставленные задачи: поставить чайник, закрыть окна, но и научилась заказывать еду и парковать машину!

Вся система состоит из набора устройств, которые связны с сервером. Сервер – мозг для «умного дома». Он может контролировать связанные с ним устройства, собирать информацию и представлять её для человека в том виде, в котором она будет удобна для анализа и управления.

Сервер будет реализован с использованием технологий и языков программирования таких как JavaScript, Node.js, Express, Angular 11.

Это обеспечит возможность расширять возможности сервера, не нарушая работы его существующей функциональности. Использование Angular даст возможность удобного управления сервером и компонентами системы.

Весь стек технологий использует язык JavaScript. Он пользуется большой популярностью среди разработчиков. Это даёт возможность расширять приложение другими людьми.

Node.js – среда выполнения языка JavaScript на локальной машине. Она как правило используется для создания компьютерных приложений, серверов и т.д. Для создания требуемого сервера был выбран фреймворк Express, так как он прост в изучении и использовании.

Express – JavaScript фреймворк, используемый для написания серверной части веб-приложений. Позволяет рендерить шаблон страницы на сервере или создавать API, отправляющее данные клиенту в формате JSON.

Angular – JavaScript фреймворк, используемый для создания клиентской части веб-приложений. Благодаря компонентной архитектуре, приложения, использующие этот фреймворк легко масштабируются. Angular предоставляет множество функций для оптимизации, нахождения багов, анимации, сервисы и т.д [1]. Была выбрана последняя версия фреймворка для облегчения разработчикам процесса улучшения приложения в будущем.

Ещё одним преимуществом выбора JavaScript в качестве основного языка разработки была возможность запускать код на микрокомпьютере Raspberry Pi. Плюсами такого решения являются то, что микрокомпьютер потребляет значительно меньше энергии, имеет разъем RJ-45 и возможность подключения по Wi-Fi и Bluetooth [3]. Благодаря этому, Raspberry Pi можно подключить к интернету и, следственно, использовать приложение из любой точки мира, где есть доступ в интернет.

Разработка ведётся по методологии Scrum. Она позволяет грамотно распределить задачи по времени и даёт возможность другим разработчикам подключиться к созданию или улучшению приложения на любой стадии.

В процессе разработки пишется «инлайн-документация». Это подход к написанию документации, при котором во все файлы, где необходимо, добавляется описание того, что происходит внутри. Это могут быть объяснения архитектуры классов, описания функций или примеры использования.

В качестве базы данных выбрана PostgreSQL. Она имеет множество встроенных функций и, благодаря большому сообществу разработчиков, позволяет легко найти их описание. PostgreSQL предоставляет собственную систему управление и поддерживается большинства другими системами управления [2].

В результате получаем архитектуру приложения, в которой клиент (датчик, микроконтроллер) отправляет запрос на сервер с некоторыми данными о своём состоянии), сервер записывает полученную информацию в базу данных и отправляет сообщение об успешной операции или ошибке обратно на датчик или микроконтроллер. Если в роли клиента выступает человек, то по переходу на адрес сервера, он направляется на сайт для управления датчиками и просмотра статистики.

Визуальное представление архитектуры представлены на рисунке 1.

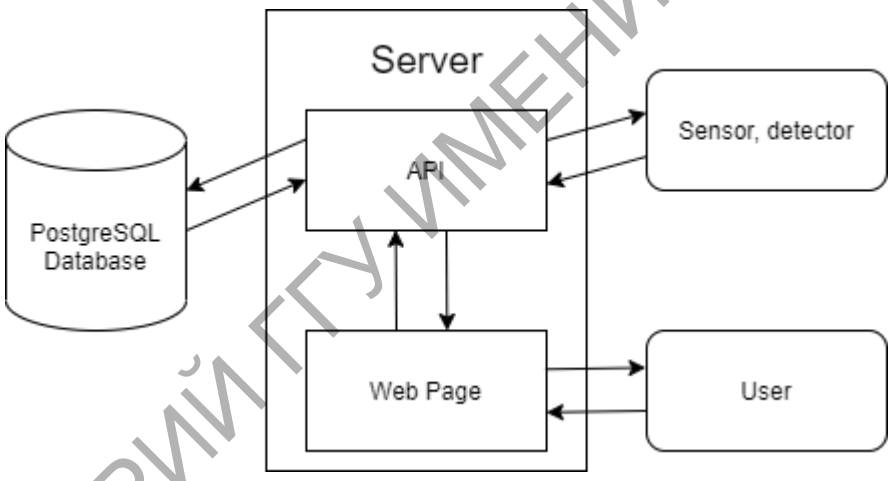

Рисунок 1 – Архитектура приложения

Предполагаемый результат – рабочий прототип сервера, использующий перечисленные ранее технологии. Главной целью разработки является создания масштабируемого приложения для управления и сбора статистики компонентов системы.

Литература

1. Документация фреймворка Angular [Электронный ресурс] – 2021. Режим доступа: https://angular.io/. – Дата доступа: 28.03.2021.

2. Документация PostgreSQL [Электронный ресурс] – 2021. Режим доступа: https://www.postgresql.org/docs/. – Дата доступа: 29.03.2021.

3. Техническая документация Raspberry Pi [Электронный ресурс] – 2021. Режим доступа: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/. – Дата доступа: 29.03.2021.

В. А. Рубин
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **С. П. Жогаль,** канд. физ.-мат. наук, доцент

СТРУКТУРИРОВАННОЕ НЕЙРОННОЕ ОБУЧЕНИЕ В TFX

Tensor Flow Extended (далее по тексту TFX) – это бесплатная платформа с открытым исходным кодом, которая предназначена для создания готовых к работе конвейеров машинного обучения. Внутри своего ядра TFX предлагает огромное количество подходов для обработки данных.

Neural Structured Learning (NSL) – это библиотека в TensorFlow, которую возможно использовать при обучении нейронных сетей, сигнал которых структурирован. Она может обрабатывать ввод двумя различными способами: как явный неявный граф и, соответственно, как явный граф, соседи которого динамически генерируются в процессе обучения модели. NSL с явным графом часто используют для предобучения нейронным графам, а NSL с неявным графом используют при состязательном обучении. Эти подходы реализованы в виде формы регуляризации в схеме NSL. Результатом этого является влияние только на процесс обучения и неизменность процесса обслуживания модели.

Вкратце, процесс построения модели с регуляризацией графа можно разбить на три шага:

1. Если граф еще недоступен – построить его.
2. Использовать особенности входного набора данных и граф для увеличения обучающей выборки.
3. Применить расширенные данные из предыдущего шага для регуляризации графа для данной модели.

К сожалению, сразу применить эти шага для TFX-конвейера нельзя, но TFX позволяет настраивать компоненты. И при помощи этой настройки можно обеспечить обработку своих собственных TFX-конвейеров. Для того, чтобы построить ругуляризованную модель графа при помощи TRX, можно воспользоваться настраиваемыми плагинами TFX.