Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(Национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

РАБОТА ПРОВЕРЕНА
РЕЦЕНЗЕНТ,
_____
_____
«____»_____2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ
ЗАВЕДУЮЩИЙ КАФЕДРОЙ,
Д.Ф.-М.Н., ДОЦЕНТ
_____С.А.ЗАГРЕБИНА
«____»_____2017 г.

Анализ эффективности выращивания культурных растений с использованием методов гидропоники и интернета вещей

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ– 01.03.02.2017.082.20.000 ВКР

Нормоконтролер,
доцент каф. МиКМ,
к.ф.-м.н., доцент
_____Т.А.Макаровских
_____2017 г.

Руководитель работы,
доцент каф. МиКМ,
к.т.н., доцент
_____В.И.Дударева
_____2017 г.

Автор работы
Студент группы ЕТ-485
_____С.А.Шумилов
_____2017 г.

Челябинск, 2017

Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(Национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»
Направление «Прикладная математика и информатика»

ДОПУСТИТЬ К ЗАЩИТЕ
ЗАВЕДУЮЩИЙ КАФЕДРОЙ,
Д.Ф.-М.Н., ДОЦЕНТ
_____С.А.ЗАГРЕБИНА
«____»_____2017 г.

ЗАДАНИЕ
на выпускную квалификационную работу студента
Шумилова Сергея Алексеевича
Группа ЕТ-485

1 Тема работы

Анализ эффективности выращивания культурных растений с использованием методов гидропоники и интернета вещей утверждена приказом по университету от _____ 20__г. №_____

2 Срок сдачи студентом законченной работы _____

3 Исходные данные к работе

– Черняк, Л. Платформа Интернета вещей. [Электронный ресурс] // Открытые системы. – 2012. – 4 сентября – М.: Открытые системы, 2015. – Режим доступа: https://www.osp.ru/os/2012/07/13017643/, свободный. – Загл. с экрана. (дата обращения: 18.04.2017);

– Тексье, У. Гидропоника для всех. Все о садоводстве на дому / Ульям Тексье. – М.: Альпина, 2011. – 265 с.

4 Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

4.1 Описание концепции интернета вещей применительно к управлению

гидропонными установками
4.2 Описание модели облачного сервиса по автоматизации работы гидропонных установок
4.3 Программная разработка экспериментальной версии облачного сервиса по автоматизации работы гидропонных установок
4.4 Описание и реализация модели прогнозирования сроков созревания урожая на основе математического аппарата искусственных нейронных сетей
4.5 Описание экономической и практической составляющей

5 Перечень графического материала (с точным указанием обязательных чертежей, плакатов в листах формата А1)

5.1 Актуальность
5.2 Цели и задачи
5.3 Модель облачного сервиса
5.4 Пользовательский интерфейс
5.5 Прогнозирование
5.6 Экономическая составляющая
5.7 Заключение

Всего __ листов

6 Календарный план

Наименование этапов дипломной работы	Срок выполнения этапов работы	Отметка о выполнении
Обзор концепции интернета вещей	30.01.2017 – 15.02.2017	
Описание модели облачного сервиса	16.02.2017 – 28.02.2017	
Написание программной реализации	01.03.2017 – 09.03.2017	
Разработка прогнозной модели на основе теории ИНС	10.03.2017 – 04.04.2017	
Описание экономической и практической составляющие, формулировка выводов	05.04.2017 – 20.04.2017	
Подготовка пояснительной записки дипломной работы	21.04.2017– 10.05.2017	
Оформление пояснительной записки	11.05.2017– 25.05.2017	
Получение отзыва руководителя	30.05.2017	

Проверка работы руководителем, исправление замечаний	25.05.2017 – 31.05.2017	
Подготовка графического материала и доклада	31.05.2017 – 01.06.2017	
Нормоконтроль	01.06.2017	
Рецензирование, представление зав. кафедрой	01.06.2017	

7 Дата выдачи задания_____

Заведующий кафедрой_____/С.А. Загребина/
(подпись)

Руководитель_____/В.И.Дударева/
(подпись)

Задание принял к исполнению_____/С.А.Шумилов/
(подпись студента)

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(Национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

АННОТАЦИЯ

Шумилов С.А. Анализ эффективности выращивания культурных растений с использованием методов гидропоники и интернета вещей, Институт естественных и точных наук, 2017. – 66 с., 9 ил., 2 табл., 3 прил., библиогр. список – 20 названий

В дипломной работе рассмотрена система автоматизированного управления гидропонными установками. Обоснована необходимость проведения разработок в направлении интернета вещей и облачных сервисов. Составлена программная реализация сервиса.

В работе приведены описание облачного сервиса, рассмотрен программный код сервера и пользовательского интерфейса, проанализированы возможности прикладного применения подобного сервиса.

Оглавление

ВВЕДЕНИЕ

Актуальность и проблематика исследования эффективности выращивания культурных растений с использованием методов гидропоники и интернета вещей является актуальной, поскольку производство продуктов питания является одной из базовых и необходимых сфер деятельности человека, а его автоматизация и внедрение интернета-вещей позволит увеличить обороты производства. По данным Росстата удельный вес организаций, использующих специальные программные средства для управления автоматизированным производством и/или отдельными техническими средствами и технологическими процессами на 2015 год, составил 15,1% от общего числа обследованных организаций. Автоматизация наряду с децентрализацией могут повысить рентабельность работы предприятий по выращиванию агрокультур. Понижение входного порога является важной частью распространения технологий для рядовых пользователей, что в свою очередь может повысить качество питания в электрифицированных, но отдаленных регионах России.

В данной работе была спроектирована и рассмотрена концепция облачного сервиса для распределенного сбора и анализа данных, управления гидропонными установками по выращиванию культурных растений, а также реализован программный интерфейс для анализа эффективности роста растений. Концепция сервиса базируется на теории интернета вещей и методиках выращивания растений без грунта. Для анализа данных применялся математический аппарат искусственных нейронных сетей.

Объектом исследования являются гидропонные установки по выращиванию культурных растений.

Предметом исследования является система автоматизированного управления гидропонными установками.

Целью работы являются разработка концепции и реализация экспериментальной версии облачного сервиса по автоматизации работы гидропонных установок.

В работе поставлены и решены следующие ***задачи:***

- описание формы построения сервисов – SaaS (англ. software as a service – программное обеспечение как услуга);
- описание современных методов построения систем интернета вещей;
- описание методов без грунтового выращивания агрокультур;
- описание математического аппарата искусственных нейронных сетей;
- реализация демонстрационной версии облачного сервиса;
- создание публичного программного интерфейса сервиса;
- создание демонстрационной версии прогнозной системы на основе ИНС (Искусственная нейронная сеть);
- реализация программного обеспечения для микроконтроллера управляющего работой гидропонной установки;
- создание пользовательского интерфейса для сервиса, работающего на базе публичного программного интерфейса.

Информационной базой работы являются данные с официального сайта Российского исследовательского и консалтингового центра Internet of things, статьи таких авторов как Леонид Черняк, Кевин Эштон, Хенг ЛеХонг и Дейв Эванс.

Работа состоит из введения, четырех глав, заключения, библиографического списка и приложения. Объем работы составляет 66 страниц, объем библиографии – 20 источников.

В первой главе работы приводится описание базовых элементов системы. Проводится анализ предметной области, общее описание гидропоники. Анализируются преимущества использования интернет вещей, а также рассматриваются три основных модели обслуживания облачных решений. Приводится общее описание кроссплатформенности веб-приложения.

Во второй главе сформировано общее описание логических частей модели облачного сервиса, приведены схемы взаимодействия между частями системы, рассматриваются принципы работы микроконтроллера, приводится представление пользовательского интерфейса, а также описание серверной части

и её задач. Рассматривается способ прогнозирования через работу с искусственной нейронной сетью.

Третья глава посвящена реализации облачного сервиса, приводится сравнение различных типов микроконтроллеров, приводится состав датчиков и предлагаемый состав управляющих устройств, приведены примеры различных функций. Также проводится обзор исторической части в приложениях. Отдельно выделена реализация серверной части и описание программного интерфейса, использование искусственной нейронной сети для реализации прогнозирования степени готовности урожая. В конце главы приводится изображение и описание пользовательского интерфейса сервиса.

В четвертой главе приводится технико-экономическое обоснование сервиса, описаны его возможные способы прикладного использования на предприятиях и частными лицами. Приведен подсчет себестоимости автоматизации частной гидропонной установки.

В заключении формулируются выводы по дипломной работе, оценивается наличие достижения цели, степень выполнения поставленных задач.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание интернета вещей

Интернет вещей – это концепция физических объектов (вещей), объединённых в вычислительную сеть (в общем случае подразумевается объединение через интернет). Основная идея заключается в приобретении обычными вещами качеств, им не свойственных, за счет взаимодействия между подобные устройствами и внешним миром.

Концепция «интернета вещей» была сформулирована еще в 1999 году при Массачусетском технологическом институте Кевином Эштоном [19] на презентации для руководства Procter & Gamble. В презентации рассказывалось о том, как всеобъемлющее внедрение радиочастотных меток сможет видоизменить систему управления логистическими цепями в корпорации [15].

Начиная с 2010-х годов наполнение концепции «интернета вещей» многообразным технологическим содержанием и внедрение практических решений для её реализации считается устойчивой тенденцией в информационных технологиях[3], прежде всего, благодаря повсеместному распространению беспроводных сетей, появлению облачных вычислений, развитию технологий межмашинного взаимодействия.

Стоит отметить, что понятие интернета вещей (Internet of Things, IoT) многозначно, поскольку включает в себя ряд других понятий и даже само по себе может варьироваться в зависимости от контекста.

В одном из определений интернет вещей представляется как сеть сетей, состоящих из уникально идентифицируемых объектов (вещей), способных взаимодействовать друг с другом без вмешательства человека, через IP-подключение. Ключевым в этом определении является независимость устройств от людей и их способность передавать данные самостоятельно. Этот критерий объясняет, почему в рынок Интернета вещей не включают такие устройства как смартфоны и планшеты[5].

Согласно мировой статистике Интернета (Internet World Stats – www.internetworldstats.com) по состоянию на июнь 2012 года Интернетом пользуются 2,4 миллиарда человек. И это лишь 34% от всего мирового населения.

В 2012 году количество подключенных к Интернету устройств превысило население планеты. Сюда входят традиционные вычислительные и мобильные устройства, а также новые промышленные и бытовые устройства, которые мы называем «вещами».

Несмотря на то, что подобное количество устройств, подключенных к Интернету, кажется ошеломляющим, это менее 1% от всех объектов, которые можно подключить.

Сегодня большое разнообразие маленьких и недорогих, но мощных датчиков можно присоединить практически к любому устройству. Эти сенсоры передают данные с заданной периодичностью (например, раз в секунду) и некоторой точностью. Результатом этой генерации данных является серия равномерных по времени точек со значениями для каждой из них.

1.2 Разработка устройств на основе концепции интернета вещей

На самом высоком уровне абстракции можно разделить интернет вещей на программную и аппаратную составляющие. В свою очередь программная составляющая делится на следующие части:

1) ПО для конечного исполнительного устройства («вещи»);

2) ПО для устройств/каналов связи, обеспечивающих коммуникацию «вещей» между собой;

3) ПО, непосредственно обеспечивающее полезную нагрузку (функциональность), т.к. сама по себе возможность устройств общаться между собой не несёт пользы для человека. Часто подразумевается, что эта часть должна быть реализована в облаке.

Аппаратную составляющую также можно разделить на подобные группы:

1) платформа для конечного исполнительного устройства («вещи»);

2) устройства и каналы связи, обеспечивающие коммуникацию «вещей» между собой;

3) облачные решения, которые иногда, реализуются на базе уже существующих, для этого достаточно реализовать шлюзы, что отлично вписывается в концепцию.

Сложность концепции интернета вещей заключается в том, что необходимо реализовать одновременно все уровни и этапы системы.

В наше время технология нашла множество применений в быту и на производстве в частности можно отметить вклад, внесенный в продвижение высокоавтоматизированных вертикальных ферм, в которых также используется метод гидропоники.

1.3 Описание способа выращивания растений методом гидропоники

Гидропоника – это система выращивания и культивирования растений без использования почвы. Роль поставщика всех необходимых растению элементов в этом случае играет водный раствор. Причём для каждой группы растений такой раствор подбирается индивидуально.

Проведя анализ способов выращивания овощных культур в защищенном грунте можно сделать заключение о том, что наиболее перспективным по рационально используемой территории, снижению затрат труда, снижению себестоимости продукции является способ выращивания на малообъёмной гидропонике.

Малообъемная технология выращивания овощей в теплицах предусматривает создание оптимальных водно-воздушных, питательных, температурных параметров в корнеобитаемой зоне растений, которая сокращена до 2 – 15 л субстрата на одно растение.

Для автоматизации данного метода выращивания растений, с помощью компьютерных средств и концепции интернета вещей собирается большое количество данных. Эффективно использовать такие объемы помогает наука о данных.

1.4 Большие данные

Мировой объем оцифрованной информации растет по экспоненте. По данным компании IBS, к 2003 году мир накопил 5 эксабайтов данных (1 ЭБ = 1

млрд гигабайтов). К 2008 году этот объем вырос до 0,18 зеттабайта (1 ЗБ = 1024 эксабайта), к 2011 году – до 1,76 зеттабайта, к 2013 году – до 4,4 зеттабайта. В мае 2015 года глобальное количество данных превысило 6,5 зеттабайта. К 2020 году, по прогнозам, человечество сформирует 40 – 44 зеттабайтов информации.

По расчетам IBS, в 2013 году только 1,5% накопленных массивов данных имело информационную ценность.

Приближается момент, когда трафик, сгенерированный датчиками и умными устройствами, превысит интернет-трафик, сгенерированный людьми.

Big data (большие данные) – огромные объемы неоднородной и быстро поступающей цифровой информации, которые невозможно обработать традиционными инструментами.

В русскоязычной среде под большими данными подразумевают также технологии их обработки.

Термин big data введён в 2008 году. Редактор журнала Nature Клиффорд Линч употребил это выражение в спецвыпуске, посвященном взрывному росту мировых объемов информации. Хотя, конечно, сами большие данные существовали и ранее. По словам специалистов, к категории big data относится большинство потоков данных свыше 100 Гб в день.

Анализ больших данных позволяет увидеть скрытые закономерности, незаметные ограниченному человеческому восприятию. Это дает беспрецедентные возможности оптимизации всех сфер нашей жизни: государственного управления, медицины, телекоммуникаций, финансов, транспорта, производства и так далее [9].

На текущий момент происходит переход количества в качество. Собираемых данных стало очень много, в том числе из-за значительного снижения стоимости датчиков. Резко упала стоимость хранения данных. В соответствии с законом Мура устойчиво продолжается увеличение мощности компьютеров при снижении их стоимости. Вычислительные мощности, которые обходились в 1955 г. в 10 долл., в 1965-м стоили 10 центов и стоят одну миллиардную цента сегодня.

В результате стало возможным достаточно дешево собирать и хранить огромные объемы информации и перейти на новый уровень детализации при ее обработке. Раньше данные, зачастую собираемые вручную, были дорогими. Их было мало, и оценки проводились по выборке из группы.

1.5 Нейронные сети

Другой эффект действия закона Мура — вернувшийся интерес к самообучающимся системам и системам с искусственным интеллектом (ИИ). Стремительно возросшие вычислительные мощности привели к тому, что многие задачи стало возможным решать простым перебором. Большой прогресс достигнут в области самообучающихся систем. Программа AlphaGo, разработанная компанией Google DeepMind, выиграла в игру Го у профессионала высшего дана. При этом она использовала недавние успехи в области машинного обучения, а именно глубинное обучение (Deep Learning) с помощью многоуровневых нейронных сетей.

Нейронные сети – это адаптивные системы для обработки и анализа данных, которые представляют собой математическую структуру, имитирующую некоторые аспекты работы человеческого мозга и демонстрирующие такие его возможности, как способность к неформальному обучению, способность к обобщению и кластеризации неклассифицированной информации, способность самостоятельно строить прогнозы на основе уже предъявленных временных рядов. Главным их отличием от других методов, например, таких, как экспертные системы, является то, что нейросети в принципе не нуждаются в заранее известной модели, а строят ее сами только на основе предъявляемой информации. Именно поэтому нейронные сети и генетические алгоритмы вошли в практику всюду, где нужно решать задачи прогнозирования, классификации, управления – иными словами, в области человеческой деятельности, где есть плохо алгоритмизируемые задачи, для решения которых необходимы либо постоянная работа группы квалифицированных экспертов, либо адаптивные системы автоматизации, каковыми и являются нейронные сети.

Нейронная сеть принимает входную информацию и анализирует ее способом, аналогичным тому, что использует наш мозг. Во время анализа сеть обучается (приобретает опыт и знания) и выдает выходную информацию на основе приобретенного ранее опыта.

Основная задача аналитика, использующего нейронные сети для решения какой-либо проблемы, – создать наиболее эффективную архитектуру нейронной сети, т.е. правильно выбрать вид нейронной сети, алгоритм ее обучения, количество нейронов и виды связей между ними. Эта работа не имеет формализованных процедур, она требует глубокого понимания различных видов архитектур нейронных сетей, включает в себя много исследовательской и аналитической работы, и может занять достаточно много времени.

Для неформализованных задач нейросетевые модели могут на порядок превосходить традиционные методы решения. Но применение нейронных сетей целесообразно, если:

- накоплены достаточные объемы данных о предыдущем поведении системы;
- не существует традиционных методов или алгоритмов, которые удовлетворительно решают проблему;
- данные частично искажены, частично противоречивы или не полны и поэтому традиционные методы выдают неудовлетворительный результат.

Нейронные сети наилучшим образом проявляют себя там, где имеется большое количество входных данных, между которыми существуют неявные взаимосвязи и закономерности. В этом случае нейросети помогут автоматически учесть различные нелинейные зависимости, скрытые в данных. Это особенно важно в системах поддержки принятия решений и системах прогнозирования.

Нейронные сети все чаще применяются в реальных бизнес приложениях. В некоторых областях, таких как обнаружение фальсификаций и оценка риска, они стали бесспорными лидерами среди используемых методов. Их использование в

системах прогнозирования и системах маркетинговых исследований постоянно растет.

Стоит отметить, что поскольку экономические, финансовые и социальные системы очень сложны и являются результатом действий и противодействий различных людей, то является очень сложным (если не невозможным) создать полную математическую модель с учетом всех возможных действий и противодействий. Практически невозможно детально аппроксимировать модель, основанную на таких традиционных параметрах, как максимизация полезности или максимизация прибыли.

В системах подобной сложности является естественным и наиболее эффективным использовать модели, которые напрямую имитируют поведение общества и экономики. А это как раз то, что способна предложить методология нейронных сетей.

1.6 Облачные сервисы интернета вещей

Основными преимуществами облачных решений можно считать, то что распределённая система не имеет единой точки отказа, но, как правило, имеет одну точку входа (общий интерфейс) для доступа к функциональности. За счёт этого обеспечивается надёжность, которая включает в себя следующие факторы:

- доступность сервиса;
- надёжность хранения данных.
- удобство его эксплуатации за счёт использования общего интерфейса и отсутствия необходимости настройки [12].

В целом можно разделить облачные сервисы по типу функциональности на облачные вычисления и облачные хранилища данных. Для интернета вещей интерес составляют оба этих решения.

Как правило доступность сервиса определяется временем в процентном соотношении от общего расчётного периода (как правило, года). Также часто применяется метрика с количеством девяток в проценте доступности (от 1 до 9), характеризующее то же самое, но в более удобной форме. Для резервирование

данных применяют самые различные системы, зачастую разнесённые территориально, чтобы снизить влияние внешних факторов на датацентры, а также, чтобы понизить время доступа из любой точки мира.

Надёжность хранения и доступность сервиса реализуется за счёт резервирования/дублирования всех составляющих: накопителей, вычислителей и каналов связи. Это является кардинальным отличием от ячеистых сетей, где резервируются только каналы связи.

Существует 3 наиболее распространённых модели обслуживания облачных решений:

- IaaS – инфраструктура как услуга (англ. Infrastructure-as-a-Service);
- PaaS – платформа как услуга (англ. Platform-as-a-Service);
- SaaS – программное обеспечение как услуга (англ. Software-as-a-Service).

Отличаются они предоставляемой функциональностью и, соответственно, возможностями, а также, как следствие, требуемыми ресурсами на разработку полного решения. Рассмотрим их по возрастанию уровня абстракции:

Инфраструктура как услуга (IaaS)

Модель «инфраструктура как услуга», сокращенно IaaS, включает в себя базовые элементы для построения облачной ИТ-системы. В рамках этой модели пользователи получают доступ к сетевым ресурсам, к виртуальным компьютерам или выделенному аппаратному обеспечению, а также к хранилищам данных. Модель «инфраструктура как услуга» обеспечивает наивысший уровень гибкости эксплуатации и управления ИТ-ресурсами. Она практически аналогична современной модели ИТ-ресурсов, привычной для персонала ИТ-отделов и разработчиков.

Платформа как услуга (PaaS)

Модель «платформа как услуга» не требует от организации управления базовой инфраструктурой (обычно включающей оборудование и операционные системы) и позволяет посвятить все усилия разработке и управлению приложениями. Это повышает производительность работы, поскольку это

избавляет от необходимости беспокоиться о приобретении материально-технических ресурсов, заниматься планированием мощности, обслуживанием ПО, установкой обновлений безопасности и выполнять другие трудоемкие задачи, необходимые для работы приложений.

Программное обеспечение как услуга (SaaS)

В рамках модели «программное обеспечение как услуга» пользователь получает завершенный продукт, работающий под управлением поставщика услуги. Обычно в этом случае речь идет о приложениях для конечных пользователей. При работе с моделью SaaS нет необходимости беспокоиться о поддержке сервиса или управлении базовой инфраструктурой и можно полностью сконцентрироваться на использовании определенного программного обеспечения. Всем известный пример приложения SaaS – веб-сервис электронной почты, позволяющий отправлять и получать электронные письма без необходимости управлять дополнениями к программному продукту или обслуживать серверы и операционные системы, на которых работает сервис [10].

1.7 Общее описание кроссплатформенности веб-приложений

Для обеспечения пользователей доступом к сервису требуется реализовать интерфейс доступа. В качестве такого интерфейса может быть использован доступ по публичному программному интерфейсу (application programming interface – API). Обычно поверх API создаются приложения, обрабатывающие данные и предоставляющие данные в удобном для взаимодействия виде [20].

На текущий момент все приложения можно разделить на ПО, написанное специально для конкретной платформы или веб-приложения. Такое разделение будет корректно как для персональных компьютеров, так и для мобильных операционных систем.

Использование любого из вариантов может быть актуально, исходя из задач и ресурсов. В случае написания облачного сервиса отсутствуют специальные требования по производительности клиентской части, и в тоже время есть необходимость в кроссплатформенности. Хорошим решением будет использование веб-приложений.

Специализированное ПО может получить все доступные на устройстве ресурсы, а значит потенциально такое ПО является более производительным.

Основным недостатком является отсутствие кроссплатформенности. Если требуется написание приложения под несколько операционных систем, то разработка такого ПО будет более ресурсоемкой.

Веб-приложения являются полным антиподом заточенных под конкретную систему программ. Приложения, написанные на веб-технологиях, имеют практически полную кроссплатформенность. К недостаткам относится невысокая, за счет использования браузера в качестве прослойки, производительность программ.

Вывод по главе один

Облачный сервис по автоматизации выращивания культурных растений с помощью гидропонных установок является объединением разных концепций, теорий и технологий, таких как: интернет вещей, гидропоника, облачные вычисления, и т.д.

Об актуальности создания подобного облачного сервиса можно судить исходя из популярности использования подобных сервисов на уровне предприятий или компаний. Добавление в подобную систему автоматизации интернет составляющей, открывает возможность получения большого количества данных, которые в свою очередь являются ключевым элементом для прогнозной модели. Также за счет использования технологий облачного вычисления повышается уровень отказоустойчивости и надежности системы в целом.

2 ОПИСАНИЕ ОБЛАЧНОГО СЕРВИСА

Модель сервиса можно разделить на три логические части:

1) программное обеспечение для микроконтроллера, отвечающее за управление гидропонной установкой;

2) пользовательский интерфейс, предоставляемый в формате электронной страницы, позволяющий следить за показателями датчиков и прогнозами системы;

3) сервер, отвечающий за обработку запросов от микроконтроллера и пользователя.

2.1 Микроконтроллер

Микроконтроллер отвечает за:

- сбор данных с варьируемого числа датчиков;
- автоматизацию включения/выключения света, поддержание установленной длины светового дня;
- обмен данными с сервером.
- корректировку питательного раствора, в соответствии с указаниями сервера.

Связь с сервером устанавливается посредством HTTP-протокола, который служит каналом передачи данных. Микроконтроллер пересылает полученную с датчиков информацию в формате, понимаемом сервером. (рисунок 1)

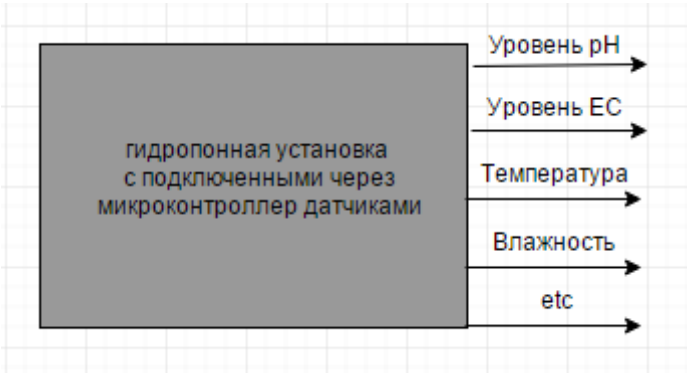

Рисунок 1 – Схема работы МК

В ответ сервер посылает управляющие команды, сообщая, как и насколько нужно откорректировать состояние установки. (рисунок 2)

Рисунок 2 – Схема работы МК

2.2 Пользовательский интерфейс

Пользовательский интерфейс может представлять из себя как веб-сайт, так и мобильное приложение. Задачей пользовательского интерфейса является предоставление клиенту данных в удобном для понимания виде: наглядные значения и графики. Также пользовательский интерфейс предоставляет возможность ручного управления частью параметров гидропонной установки. Схема работы пользовательского интерфейса приведена на рисунке 3.

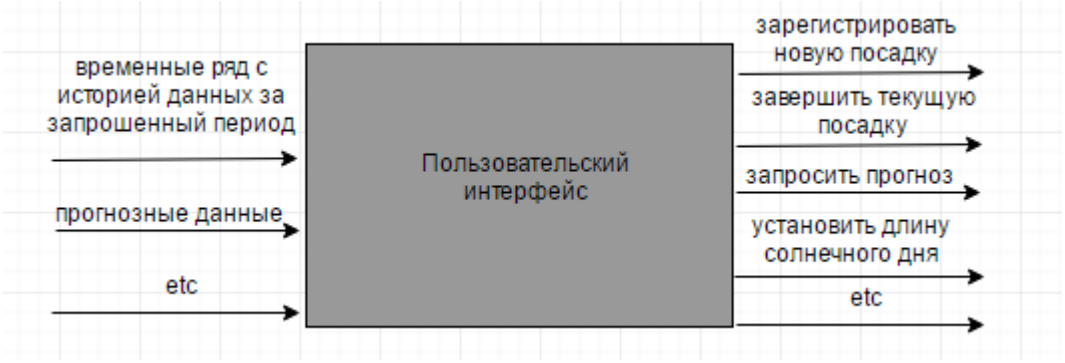

Рисунок 3 – Схема работы пользовательского интерфейса

2.3 Серверная часть

Сервер выполняет роль связующего звена и центра управления, в его задачи входит:

1) предоставление доступа к функциональности сервиса по публичному программному интерфейсу;

2) обработка запросов от пользователей и микроконтроллеров;

3) анализ полученных от микроконтроллера данных и ответ корректирующими указаниями;

4) построение прогнозных данных;

5) сохранение консистентности внутри базы данных.

Схема работы сервера приведена на рисунке 4.

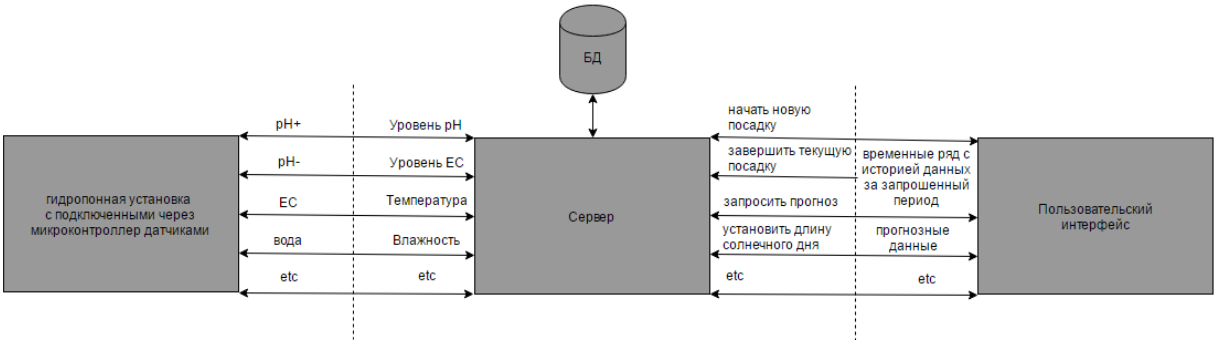

Рисунок 4 – Схема работы сервера

2.4 Описание способа прогнозирования

Для прогнозирования срока до созревания урожая была выбрана математическая модель искусственной нейронный сети(ИНС) типа перцептрон, с шестью входными и одним выходным значениями. Выбор обусловлен хорошей масштабируемостью и приемлемой точностью данного метода прогнозирования.

Входные параметры:

1) количество дней с начала проращивания;
2) средний уровень pH за сутки;
3) средний уровень EC за сутки;
4) средний уровень температуры воды за сутки;
5) средний уровень температуры воздуха за сутки;
6) в зависимости от текущего дня: если первый день, то ожидаемое производителем количество дней до созревания, иначе – результат работы ИНС для предыдущего дня.

Единственным выходным значением является прогнозируемое количество дней до созревания.

Для корректной работы ИНС важны объем и полнота выборки данных, т.к. на стадии разработки не существует фактических данных в требуемом формате, данные для начального обучения ИНС будут генерироваться. Легкая масштабируемость системы прогнозирования позволит с необходимой скоростью

отказаться от генерации данных для обучения ИНС и со временем использовать только данные, получаемые от пользователей.

Выводы по главе два

Описанная модель сервиса является общим решением и имеет большую степень вариативности, так, например, можно представить вариант сервиса не имеющего прямой связи с микроконтроллерами, а взаимодействующего только с пользователем.

3 РЕАЛИЗАЦИЯ ОБЛАЧНОГО СЕРВИСА

3.1 Реализация работы микроконтроллера

Для реализации гидропонной установки выбрана описанная в главе 1 система глубоководных культур.

Для получения информации и управления гидропонной установкой требуется выбрать микроконтроллер. Для поставленных задач подойдут как одноплатные компьютеры, так и платы на микроконтроллере. Типичными представителями одноплатных компьютеров являются Raspberry Pi или Intel Edison. Среди обычных микроконтроллеров популярным решением является Arduino Uno.

Сравнение трех систем приведено в таблице 1.

Таблица 1 – Сравнение микроконтроллеров

	Arduino Uno	Raspberry Pi	Intel Edison
Цена	1800 руб.	3500 руб.	6300 руб.
Сложность разработки	Наиболее популярная система, множество документации и написанного кода. Разработка осущесвляется на C/C++ подобном языке программирования	В основном, работает на операционных системах, основанных как на Linux ядре. Также возможна установка Windows 10 IOT. Любые языки программирования поддерживаемые ОС.	Среди доступных языков программирования: C, C++, Python, or JavaScript (Node.js). Также совместима с платформой Arduino.
Доступность датчиков	Множество датчиков, как от официальных производителей так и любительских.	Все датчики, которые могут понадобится являются вполне распространёнными.	Система совместима с большинством датчиков от Arduino.
Цена датчиков	Большая вариативность цен.	Средняя цена и качество несколько выше чем у Arduino.	

Исходя из сравнения, можно сделать вывод о том, что все устройства могут использоваться в качестве микроконтроллера. Делаем выбор в пользу Arduino Uno исходя из цены.

Предлагаемый состав датчиков:

1) Датчик уровня кислотности – pH;
2) Датчик измерения общего солесодержания, для измерения концентрированности питательного раствора;
3) Датчик температуры воды;
4) Датчик температуры и влажности воздуха;
5) Датчик уровня воды.

Предлагаемый состав управляющих устройств:

1) Wi-Fi модуль для установления соединения с сервером;
2) Часы реального времени для установки длины светового дня;
3) Модуль с электромеханическим реле для управления освещением;
4) Четыре погружные помпы для управления количеством и составом воды.

Общение с сервером предполагается в формате GET-запросов по адресу /api/addRow. Функция формирующая запрос к серверу приведена в листинге 1.

```
String readSensors() {
    ferm.waterLevel = getWaterLevel();
    String line = "addRow?water=" + ferm.waterLevel;
    ferm.ph = getPHLevel();
    line += "&ph=" + ferm.ph;
    ferm.temperature = getTemperature();
    line += "&temperature=" + ferm.temperature;
    ferm.temperature = getTemperature();
    line += "&waterTemperature=" + ferm.temperature;
    ferm.humidity = getHumidity();
    line += "&humidity=" + ferm.humidity;
    return line;
}
```

Листинг 1 – Функция МК, выполняющая сбор данных с датчиков

1) water – число, отражающее на сколько миллиметров от максимума опустился уровень воды в контейнере;
2) ph – текущий уровень pH;
3) ppm – концентрация питательного раствора в миллионных долях;

4) temperature – температура воздуха в градусах Цельсия;

5) waterTemperature – температура воды в градусах Цельсия;

6) humidity – влажность воздуха в процентах.

Сервер в ответ высылает управляющие данные в формате JSON (листинг 2).

```
{
    water: Number,
    “ph+”: Number,
    “ph-“: Number,
    ppm: Number,
}
```

Листинг 2 – Схема данных, получаемая от сервера

Где:

1) water – число, означающее на сколько миллисекунд необходимо включить помпу перекачивающую чистую воду;

2) ph+ и ph- – отвечает за помпы с концентратами pH+ и pH- соответственно;

3) ppm – концентрированное удобрение.

Необязательными полями является:

1) dayStart и dayEnd – поля, отвечающие за переназначение начала и конца светового дня

2) fanPower – мощность вентилятора.

3.2 Сервер

Серверная часть может быть реализована как посредствам HTTP-сервера, так и на базе популярных, для интернета вещей, облачных решений. Лидерами среди последних являются такие продукты как: ThingWorx, Iridium, Azure IoT Hub, Google Cloud.

В качестве альтернативы готовым решениям был выбран язык программирования JavaScript и программная платформа Node.js. Выбор обусловлен низким порогом входа для написания программ, большим количеством примеров и отсутствием специфичных требований к языку программирования.

Все готовые решения являются платными продуктами, а также вызывают проблему «Привязки к поставщику», т.е. сменить готовое решение будет

достаточно дорогим как по времени, так и по другим ресурсам процессом. В связи с этим, для текущей версии, был выбран вариант с написанием HTTP-сервера на языке программирования JavaScript.

Для долговременного хранения данных сервер должен иметь доступ к базе данных. Для текущей реализации выбор базы данных не обуславливается специфическими требованиями, поэтому в качестве решения для быстрой разработки была выбрана нереляционная база данных MongoDB.

3.2.1 Программный интерфейс

В рамках сервера реализован программный интерфейс, обрабатывающий запросы по адресам:

1) /api/addPlant
2) /api/completePlant
3) /api/addRow
4) /api/getStats

/api/addPlant создаёт новую итерацию выращивания растения со значениями по умолчанию. Схема базы данных для записи итерации выращивания растения показана на листинге 3.

```
{
    active: Boolean,
    recomendedDays: Number,
    name: String,
    waterCapacity: Number,
    recomendedPh: Number,
    recomendedPpm: Number,
    recomendedTemperature: Number,
    recomendedWaterTemperature: Number,
    lastRecalculationDate: Date,
    calculatedDays: Number,
}
```

Листинг 3 – Схема базы данных для состояния итераций

/api/completePlant– завершают текущую итерацию.

/api/addRow – Сервер при получении данных сохраняет их в базу данных

Схема базы данных для сохранения запросов от микроконтроллера показана на листинге 4.

```
{
    name: String,
    water: Number,
    ph: Number,
    ppm: Number,
    temperature: Number,
    waterTemperature: Number,
    humidity: Number,
}
```

Листинг 4 – Схема базы данных для хранения данных

После сохранения данных от микроконтроллера сервер обрабатывает пришедшие данные (листинг 5).

```
exports.correction = (stats, plant) => {
  return {
    water: ((plant.waterCapacity - stats.water) / (350 / (60 * 60 * 1000))).toFixed(),
    ppm: (plant.recomendedPpm - stats.ppm > 100) ? 500: 0,
    'ph+': (plant.recomendedPh - stats.ph > 1) ? 300: 0,
    'ph-': (plant.recomendedPh - stats.ph < 1) ? 300: 0,
    // dayStart: Number,
    // dayLength: Number,
    // fanPower: Number,
  }
};
```

Листинг 5 – Функция обработки данных полученных от МК

Затем сервер отправляет микроконтроллеру корректировочные данные в формате, приведенном в листинге 6.

```
{
    water: Number,
    “ph+”: Number,
    “ph-“: Number,
    ppm: Number,
    dayStart: Number,
    dayLength: Number,
    fanPower: Number,
}
```

Листинг 6 – Формат корректировочных данных отправляемых на МК

/api/getStats – предоставляет способ получения текущих данных. Результатом запроса будет являться JSON-файл со структурой, приведенной в листинге 7.

```
{
    plants: {
        active: Boolean,
        recomendedDays: Number,
        name: String,
        waterCapacity: Number,
        recomendedPh: Number,
        recomendedPpm: Number,
        recomendedTemperature: Number,
        recomendedWaterTemperature: Number,
        lastRecalculationDate: Date,
        calculatedDays: Number,
```

```
    },
    stats: {
        name: String,
        water: Number,
        ph: Number,
        ppm: Number,
        temperature: Number,
        waterTemperature: Number,
        humidity: Number,
    }
}
```

Листинг 7 – Схема JSON-файла, возвращаемого сервером

3.3 Реализация прогнозирования степени готовности урожая

Для начального обучения ИНС, за неимением фактических данных от пользователей, будут использоваться генерируемые данные с нормальным распределением.

На листинге 8 приведена функция генерации данных для обучения ИНС для выращивания огурцов со сроком созревания не более 100 дней.

```
function getData(i, days, j) {
    return [
        i/100,
        transform.normalizePh(norm.next(6, 0.5)),
        transform.normalizeEC(norm.next(1600, 200)),
        transform.normalizeTemperature(norm.next(23, 2)),
        transform.normalizeTemperature(norm.next(25, 3)),
        days,
    ];
}
```

Листинг 8 – Функция генерации данных для обучения ИНС

Здесь функция next(mean, dev) у объекта norm – возвращает нормально распределенное число с математическим ожиданием равным mean и средним квадратическим отклонением равным dev, а объект transform содержит функции, нормализующие генерируемые значения.

Для обучения ИНС на языке программирования JavaScript существует несколько библиотек: Synaptic, ConvNetJS, mind, Neataptic.

Ввиду отсутствия особых требований к библиотеке можно считать подходящей любую из популярных. Для использования была выбрана библиотека Synaptic, являющаяся второй по популярности и наиболее активно развиваемой на GitHub.

На листинге 9 показан код, отвечающий за обучения ИНС на генерируемых данных.

```
var synaptic = require('synaptic');
var Architect = synaptic.Architect;
var perc = new Architect.Perceptron(6, 50, 1);
for (j = 0 ; j < 2000001; j+=1) {
    let defaultDays = Math.floor(Math.random() * 20) + 50;
    let prevDays = defaultDays / 100;
    var dynamicRate =  .035;
    for (i = 0 ; i < defaultDays; i+=1) {
        prevDays = perc.activate(getData(i, prevDays, j))[0];
        const days = norm.next(defaultDays - i, 1);
        perc.propagate(dynamicRate, [days / 100]);
    }
}
```

Листинг 9 – Обучение ИНС

Методом перебора наиболее приемлемые данные показала ИНС типа перцептрон с одним скрытым слоем из 50 нейронов. Количество итераций составило 2 000 000, коэффициент скорости обучения – 0.035 [16].

Функция переоценки прогноза вызывается после отправки микроконтроллеру корректировочных данных. Прогнозные данные сохраняются в базе данных в схеме с текущей итерацией, как показано на листинге 10.

```
plant.calculatedDays = predictor([
    numOfDay, // № of day
    data.ph, // pH
    data.ppm, // EC
    data.waterTemperature, // waterTemperature
    data.temperature, // temperature
    plant.calculatedDays ? plant.calculatedDays : plant.recomendedDays / 100 // prevData
])[0];
```

Листинг 10 – Вызов функции переоценки прогноза

3.4 Реализация пользовательского интерфейса

Клиентская часть предоставляет пользователю графики различных показателей и прогнозные данные по созреванию урожая. Также через приложение имеется возможность создания новых и завершения текущих итераций выращивания агрокультур.

Интерфейс построен на базе языка разметки HTML, формального языка описания внешнего вида документа CSS, а также языка программирования JavaScript.

Для получения данных о текущем состоянии гидропонной установки, веб-интерфейс использует API (англ. application programming interface – программный интерфейс приложения) сервера, находящийся по адресу /api/getStats.

Снимок экрана с внешним видом интерфейса приведен на рисунке 5.

Рисунок 5 – Внешний вид интерфейса

Для завершения текущей итерации предусмотрена управляющая кнопка «Завершить текущую итерацию». По нажатию вызывается метод публичного интерфейса /api/completePlant. После выполнения действий текущая посадка считается завершенной, данные пересчитываются и сохраняются в базе данных. Изображение интерфейса приведено на рисунке 6.

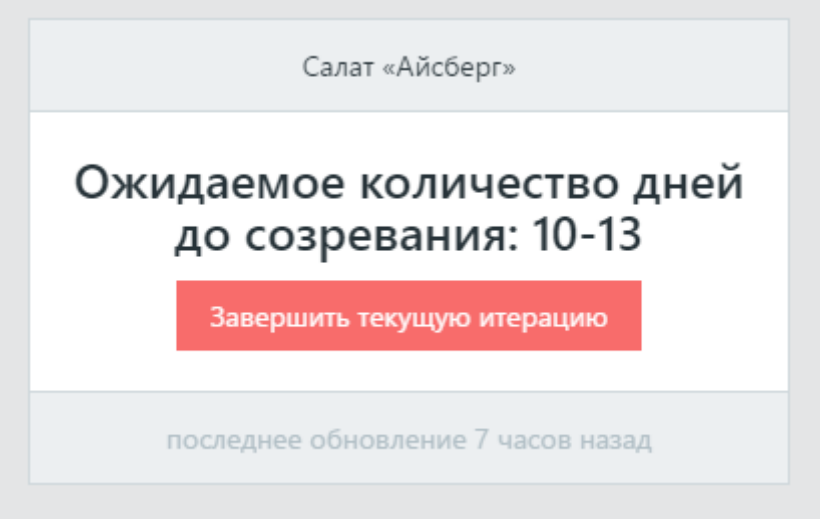

Рисунок 6 – Внешний вид интерфейса завершения итерации

Для регистрации новой посадки необходимо заполнить форму, и подтвердить выполнение действия нажатием на управляющий элемент «Начать новую итерацию». Программа выполнит запрос методу API сервера по адресу /api/addPlant, с параметрами name и recomendedDays. Здесь recomendedDays – параметр означающий ожидаемое производителями семян время выращивания. Изображение интерфейса приведено на рисунке 7.

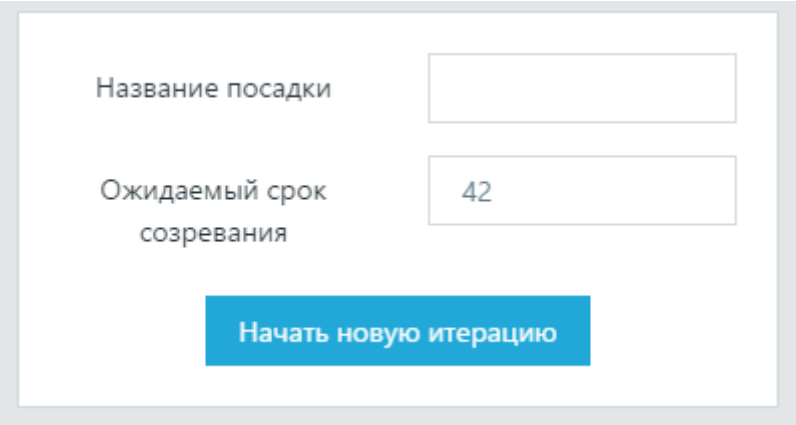

Рисунок 7 – Внешний вид интерфейса создания итерации

Выводы по главе три

Реализация тестовой версии сервиса показало, что у сервиса может быть множество имплементаций. Каждая программная часть сервиса является заменяемой и может быть улучшена на более подходящую в конкретной ситуации. Большое количество данных, предполагаемое из концепции распределённого сбора данных, позволит постоянно экспериментировать с процессом прогнозирования и находить новые зависимости.

4 ЭКОНОМИЧЕСКАЯ И ПРАКТИЧЕСКАЯ СОСТАВЛЯЮЩАЯ

Описанный комплекс программного и физического обеспечения может быть реализован в виде сервиса, клиентами которого могут стать как компании, специализирующиеся на выращивании агрокультур в промышленных масштабах, так и частные лица, покупающие овощи в магазинах или выращивающие их для собственного потребления.

Коммерческие предприятия:

Среди малых и средних предприятий можно выделить две группы производств, которых можно считать потенциальными клиентами:

1) компании, занимающиеся выращиванием агрокультур на гидропонных установках;

2) компании, выращивающие агрокультуры традиционным способом.

Компании, которые используют методы гидропоники при выращивании растений, могут быть заинтересованы в использовании специализированного облачного сервиса для сокращения расходов за счет сокращения штата людей, отвечающих за анализ и обработку данных, а также за счет полного отказа или уменьшения собственного серверного оборудования.

Также одной из причин выбора облачного сервиса взамен собственной реализации, может стать повышение отказоустойчивости посредством децентрализации серверного оборудования, а также повышение надежности системы в целом, за счет контроля бизнес-логики программного обеспечениями людьми, специализирующимися на этих задачах.

На рисунке 8 приведена возможная схема предприятия с автоматизированными гидропонными установками.

Для подключения такого производства к сервису потребуется перенаправить трафик, генерируемый их микроконтроллерами на сервера сервиса, а также написать ПО, которое позволит правильно трактовать указания сервера физическим контроллерам предприятия.

На рисунке 9 схема того же предприятия в случае перехода на использование сервиса.

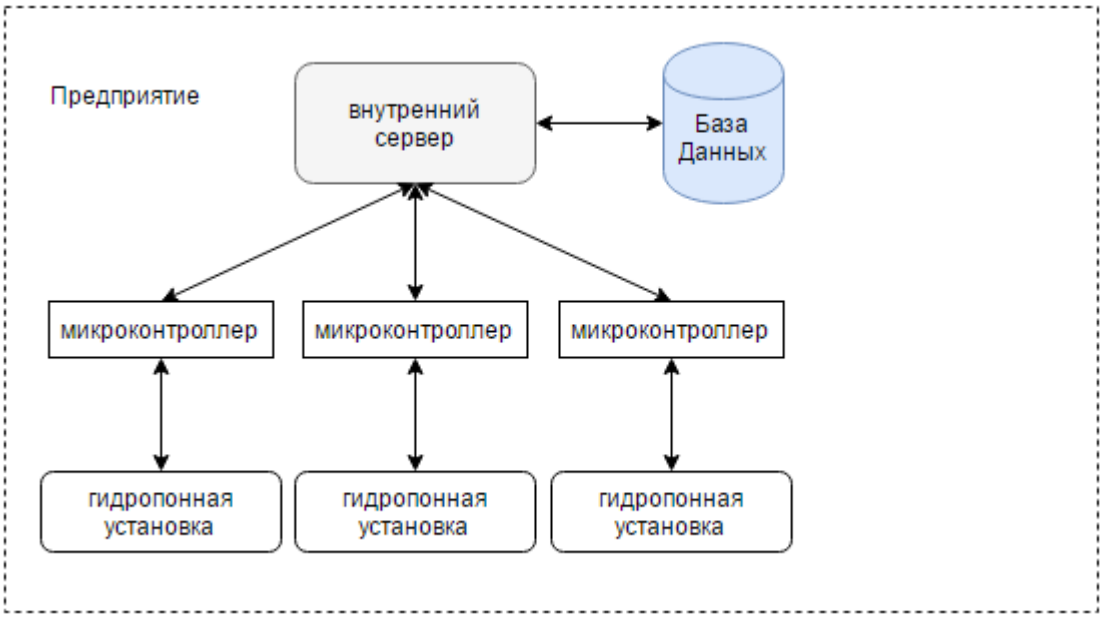

Рисунок 8 – схема предприятия с гидропонными установками

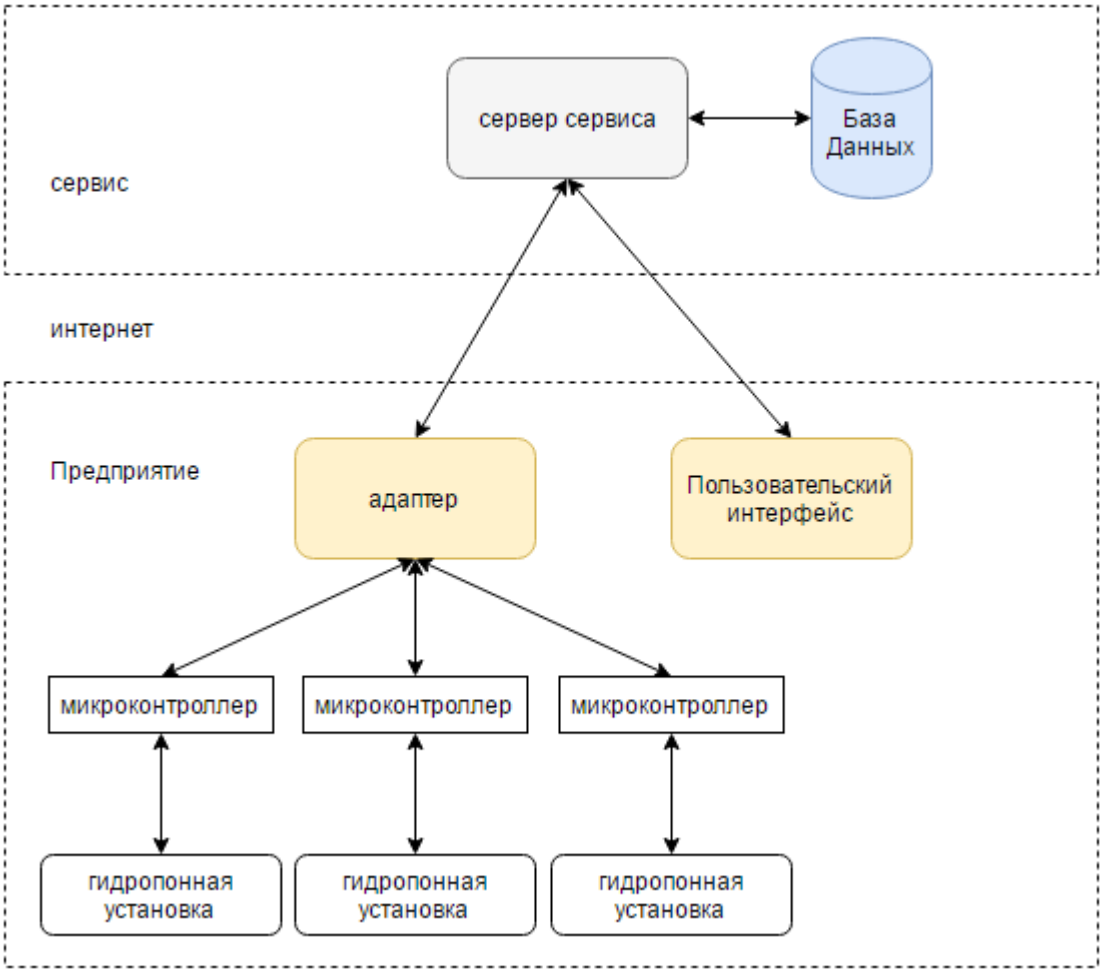

Рисунок 9 – схема предприятия, использующего облачный сервис

Использование такого сервиса позволит малому бизнесу отказаться от найма узких специалистов в области анализа данных и построения прогнозных моделей, а использовать готовые модели, предоставляемые облачным сервисом, построенные за счет большого количества данных, полученных в том числе и от конкурентов.

Компании, занимающиеся выращиванием культурных растений с помощью традиционного способа, также являются потенциальными клиентами подобного облачного сервиса.

Одна из причин отказа от перехода на гидропонный метод выращивания растений – это высокие затраты на заработную плату специалистов. На текущий момент проще и выгоднее найти людей, которые могут поливать и удобрять растения, посаженные в грунт, нежели людей с квалификацией достаточной для поддержания в актуальном состоянии серверного оборудования и анализа данных полученных с датчиков.

За счет использования облачного сервиса затраты на высококвалифицированных специалистов сокращаются и часть малых предприятий смогут отказаться от традиционного способа выращивания агрокультур в пользу гидропонного – более эффективного способа.

Частные лица:

Актуально использование для загородных домов, частного сектора, и дачных поселков.

Минимальный набор для автоматизации гидропонной установки с необходимыми для снятия показаний и управления системой датчиками может эффективно обслуживать площади размером до 10 квадратных метров. На малых площадях эффективность не будет падать, но рентабельность установки будет заметно снижена.

Расчеты стоимости минимального набора для автоматизации гидропонной системы приведены в таблице 2.

Таблица 2 –Расчеты стоимости минимального набора автоматизации

Наименование	Количество штук	Цена за штуку (рублей)
Arduino Uno	1	200
Датчик уровня кислотности – pH и датчик температуры воды	1	1000
Датчик измерения общего солесодержания	1	4000
Датчик температуры и влажности воздуха	1	300
Датчик уровня воды	1	100
Wi-Fi модуль для установления соединения с сервером	1	150
Часы реального времени для установки длины светового дня	1	100
Модуль с электромеханическим реле для управления освещением	1	100
Погружные помпы для управления количеством и составом воды	4	600

Таким образом себестоимость автоматизации гидропонной установки составит 8550 рублей.

На площади больше 10 квадратных метров объём питательного раствора в системе может достигать нескольких тысяч литров. В связи с этим встает вопрос о точности измерения как кислотности, так и общего солесодержания. Также при использовании гидропонных систем с большими объемами воды, на подобных площадях могут понадобиться более мощные помпы.

При использовании гидропонной установки с площадью, достигающей 10 квадратных метров стоимость автоматизации составит меньше 20 процентов от стоимости установки.

Для городского жителя использование облачного сервиса и автоматизации может оказаться нерентабельно в связи с малой площадью средней квартиры. При установке системы автоматизации на установку с площадью меньше одного квадратного метра, стоимость установки автоматизирующего оборудования может удвоить стоимость гидропонной установки.

Выводы по главе четыре

На основании сделанных в главе предположений можно сделать вывод о широком спектре возможной имплементации концепции облачного сервиса по автоматизации гидропонных установок. Исходя из активности рынка можно также предположить наличие большого количества потенциальных клиентов. Низкая рентабельность для малых объемов делает сервис не актуальным для физических лиц.

ЗАКЛЮЧЕНИЕ

В результате написания данной работы была спроектирована концепция облачного сервиса, предоставляющего услуги по удаленному выращиванию агрокультур. Также была реализована демонстрационная версия сервиса.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ:

1) выполнен анализ предметной области;

2) реализован пример ПО для управления гидропонной установкой на языке программирования C++, способный считывать показания датчиков, устанавливать связь с сервером по протоколу HTTP, а также управлять:

   a. набором водяных помп для добавления или корректировки питательного раствора;
   b. длиной светового дня;
   c. уровнем кислотности питательного раствора.

3) реализована демонстрационная версия облачного сервиса по автоматизации работы гидропонных установок, включающая в себя:

   a. публичный программный интерфейс для работы с разными платформами, в том числе микроконтроллерами и клиентскими программами
   b. логику обработки данных полученных от микроконтроллера;
   c. работу с базой данных для долговременного хранения результатов;
   d. демонстрационную версию прогнозной модели на основе искусственной нейронной сети, обученную на основе генерируемых модельных данных.

4) на основе веб-технологий создан пользовательский интерфейс для работы с облачным сервисом работающий на базе публичного программного интерфейса, среди возможностей которого:

   a. отображение текущей информации, полученной с датчиков в графическом виде, с помощью гистограмм;
   b. создание новой и завершение активной итерации выращивания растений в гидропонных установках.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Головко, В.А. Нейронные сети: обучение, организация и применение / В.А. Головко, А.И. Галушкин – М.: ИПРЖР, 2001. – 256 с.

2. Горбань, А.Н. Нейронные сети на персональном компьютере. / А.Н. Горбань, ДА. Россиев. – Новосибирск: Наука, 1996. – 276 с.

3. Гребнев, Е. Облачные сервисы. Взгляд из России / Е. Гребнев. – М.: CNews, 2011. – 282 с.

4. Еремин, Д.М. Искусственные нейронные сети в интеллектуальных системах управления. / Д.М. Еремин, И.Б Гарцеев. – М.: МИРЭА, 2004. – 75 с.

5. Калацкая, Л.В. Организация и обучение искусственных нейронных сетей: Экспериментальное учеб. пособие. / Л.В. Калацкая, В.А. Новиков, В.С. Садов. – Минск: Изд-во БГУ, 2003. – 72 с.

6. Клементьев, И.П. Введение в Облачные вычисления / И.П. Клементьев, Устинов В.А. – Ульяновск: УлГУ, 2009. – 233 с.

7. Круглов, В. В., Искусственные нейронные сети. Теория и практика / В. В. Круглов. – М.: Горячая линия-Телеком, 2002. – 382 с.

8. Левитин, А.В. Алгоритмы. Введение в разработку и анализ / В.А. Левитин – М.: Вильямс, 2006. – 576 с.

9. Майер-Шенбергер, В. Большие данные. Революция, которая изменит то, как мы живём, работаем и мыслим / Виктор Майер-Шенбергер, Кеннет Кукьер. – М.: Манн, Иванов, Фербер, 2014. – 240 с.

10. Модель SaaS – в мире и в России. [Электронный ресурс] URL: https://www.bytemag.ru/articles/detail.php?ID=12825 (дата обращения: 18.04.2017).

11. Осовский, С. Нейронные сети для обработки информации. / С. Осовский, И.Д. Рудинский. – М.: Финансы и статистика, 2004. – 344 с.

12. Риз, Дж. Облачные вычисления / Джордж Риз. – М.: BHV, 2011. – 288 с.

13. Тексье, У. Гидропоника для всех. Все о садоводстве на дому / Ульям Тексье. – М.: Альпина, 2011. – 265 с.

14. Черняк, Л. Большие Данные – новая теория и практика // Открытые системы. СУБД. – 2011. – № 10. – ISSN 1028-7493.

15. Черняк, Л. Платформа Интернета вещей. [Электронный ресурс] // Открытые системы. – 2012. – 4 сентября – М.: Открытые системы, 2015. – Режим доступа: https://www.osp.ru/os/2012/07/13017643/, свободный. – Загл. с экрана. (дата обращения: 18.04.2017).

16. Ashton, K. That «Internet of Things» Thing. [Electronic resource] / Kevin Ashton // RFID Journal – 2009 – Jun, 22 – Emerald Expositions, 2017 – Режим доступа: http://www.rfidjournal.com/articles/view?4986 свободный. – Загл. с экрана. (дата обращения: 18.04.2017).

17. Chen, M. Big Data. Related Technologies, Challenges, and Future Prospects. / Min Chen, Shiwen Mao, Yin Zhang, Victor C.M. Leung. – Luxembourg: Spinger, 2014. – 100 p.

18. Hersent, O. The Internet of Things: Key Applications and Protocols. / Oliver Hersnet, David Boswarthick, Omar Elloumi – Willey, 2012. – 370 c.

19. Kranenburg, R. The Internet of Things: A critique of ambient technology and the all-seeing network of RFID / Rob van Kranenburg – Pijnacker: Telstar Media, 2008. – 62 p.

20. Russell, Alex. «Progressive Web Apps: Escaping Tabs Without Losing Our Soul». [Electronic resource] / Alex Russell // Retrieved – 2015. –June, 15. Режим доступа: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/ свободный. – Загл. с экрана. (дата обращения: 18.04.2017).

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А
ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

Анализ эффективности выращивания культурных растений
с использованием методов гидропоники и интернета вещей

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
К ПРОГРАММНОМУ СРЕДСТВУ
ЮУрГУ – 01.03.02.2017.082.20.000 ТЗ ВКР

Нормоконтролер,
доцент каф. МиКМ,
к.ф.-м.н., доцент
_____Т.А.Макаровских
_____2017 г.

Руководитель работы,
доцент каф. МиКМ,
к.т.н., доцент
_____В.И.Дударева
_____2017 г.

Автор работы
студент группы ЕТ-485
_____Шумилов С.А.
«____»_____2017 г.

Челябинск, 2017

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1 ВВЕДЕНИЕ

1.1 Наименование программного изделия

Полное наименование программы – «Облачный сервис по автоматизации работы гидропонных установок». Краткое название программы – «ОСпАРГУ»

1.2 Область применения

Система предназначена для частичной или полной автоматизации процесса выращивания культурных растений, контроля состояния установки, прогнозирования сроков созревания урожая.

2 ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

2.1 Документ, на основании которого ведется разработка

Разработка ведется на основании задания выпускной квалификационной работы.

2.2 Организация, утвердившая этот документ, и дата его утверждения.

Задание утверждено руководителем ВКР доцентом кафедры МиКМ, кандидатом технических наук Дударевой В.И.

2.3 Наименование темы разработки

Наименование темы разработки – Программная реализация облачного сервиса по автоматизации работы гидропонных установок, построенных по концепции интернета вещей.

3 НАЗНАЧЕНИЕ РАЗРАБОТКИ

Разработка является частью дипломной работы.

4 ТРЕБОВАНИЯ К ПРОГРАММЕ

4.1 Требования к функциональным характеристикам

4.1.1 Состав выполняемых функций

4.1.1.1 Комплекс программ должен реализовывать клиентскую и серверную часть облачного сервиса.

4.1.1.2 При загрузке программного сценария в микроконтроллер Arduino, МК должна начать собирать и отправлять данные на сервер.

4.1.1.3 При запуске сервера должен начать отвечать по порту 80, для принятия запросов от клиентских приложений, в частности Arduino и пользовательского приложения.

4.1.2 Организация входных и выходных данных

Сервер взаимодействует с клиентскими приложениями посредствам пересылка данных в формате JSON. При получении от пользователя GET запроса по адресу /api/getStats приложение должно вернуть JSON-файл с данными для последующей обработки пользовательским приложением и представлении в формате HTML5.

В процессе работы программ входной информацией должны являться:

- Для сервера – POST и GET запросы по протоколу http;
- Для программной части Arduino – ответы сервера и информация с датчиков;
- Для пользовательского приложения – действия, предпринимаемые пользователем, посредствам нажатия на управляющие элементы внутри интерфейса.

4.2 Требования к надежности

4.2.1 Требования к надежному функционированию

Программа должна нормально функционировать при бесперебойной работе ЭВМ. При возникновении сбоя в работе аппаратуры восстановление нормальной работы программы должно производиться после: перезапуска исполняемого файла программы или переподключения микроконтроллера.

Уровень надежности программ должен соответствовать технологии программирования, предусматривающей:

1) инспекцию исходных текстов программы;

2) автономное тестирование модулей программы;

3) тестирование сопряжений модулей программы;

4) комплексное тестирование программы.

4.2.3 Время восстановления после отказа

Время восстановления после отказа должно состоять из:

- для сервера: времени автоматического перезапуска сервера;
- для микроконтроллера: времени ручной перезагрузки пользователем;

4.3 Условия эксплуатации

Программы могут хранится в виде исполняемых файлов с необязательной маркировкой.

4.4 Требования к составу и параметрам технических средств

Сервер должна корректно работать на персональном компьютере или серверная станция с процессором Intel i3 и выше, или совместимом с ним оборудовании.

Микроконтроллер – Arduino UNO или выше, с корректно подключенными портами.

4.5 Требования к информационной и программной совместимости

4.5.1 Требования к информационным структурам на входе и выходе

Требования к информационным структурам на входе и выходе определены в п.4.1.2.

4.5.2 Требования к языкам программирования

Язык программирования выбирается разработчиком без согласования с заказчиком.

4.5.3 Требования к программным средствам, используемым программой

- Для сервера необходима операционная система совместимая с программной платформой Node.js;
- Для пользовательского приложения необходима операционная система совместимая с любым браузером, поддерживающим HTML5.

5 ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Состав программной документации должен включать следующие документы:

1) технический проект программы по ГОСТ 19.404-79 в машинописном исполнении;

2) описание программы по ГОСТ 19.402-78 на компакт-диске;

3) текст программы по ГОСТ 19.401-78 на компакт-диске;

4) руководство пользователя по ГОСТ 19.504-79 на компакт-диске в виде файла README.TXT.

Пояснительная записка «Технический проект программы» должна содержать следующие разделы.

1) Раздел «Входные данные» (характер, организация входных данных).

2) Раздел «Выходные данные» (характер и организация выходных данных).

3) Раздел «Описание логической структуры» при технологии структурного программирования должен включать следующие материалы:

• описание связей программы с другими программами;

• описание внутренних массивов и переменных, которые используются в межмодульном обмене данными;

• схема иерархии программы (приводится рисунок);

• расшифровка наименования модулей (приводится таблица с перечнем наименований модулей в алфавитном порядке с указанием выполняемой каждым модулем функции);

• описание функционирования программы с учетом её модульного деления (приводится словесное описание выполнения программы с учетом вызовов модулей);

• описание модулей программы (подраздел заполняется на основе паспортов модулей).

4) Раздел «Используемые технические средства» (типы ПК, на которых возможно выполнение программы; устройства, используемы при выполнении программы).

5) Раздел «Вызов и загрузка» (виды носителей программы, их используемый объем; способы вызова программы с соответствующих носителей данных; входные точки в программу – запуск программы).

6) Раздел «План мероприятий по разработке и внедрению программы» (план мероприятий разрабатывается для реализации программы коллективом

программистов; планом должны быть предусмотрены контрольные временные точки реализации; приводится состав тестов и принципы для каждой из контрольных точек).

6 ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

Технико-экономические показатели должны определяться заказчиком без участия исполнителя.

7 СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Разработка программы должна выполняться по следующим этапам:

1) разработка, согласование и утверждение технического проекта программы с пояснительной запиской – 5 недель;

2) разработка рабочего проекта программы с комплексным тестированием – 6 недель;

3) приемка-сдача с исправлением обнаруженных недостатков в программе и программной документации – 2 недели.

4) внедрение.

8 ПОРЯДОК КОНТРОЛЯ И ПРЕМКИ

8.1 Виды испытаний

Испытания программы и верификация документации должны производиться в организации заказчика с привлечением сторонних экспертов. Проверочные тесты должны готовиться заказчиком.

8.2 Общие требования к приемке

Приемка программы должна осуществляться заказчиком. Программа должна считаться годной, если она удовлетворяет всем пунктам данного технического задания.

ПРИЛОЖЕНИЕ Б
РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

Анализ эффективности выращивания культурных растений
с использованием методов гидропоники и интернета вещей

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ
К ПРОГРАММНОМУ СРЕДСТВУ
ЮУрГУ – 01.03.02.2017.082.20.000 РП ВКР

Нормоконтролер,
доцент каф. МиКМ,
к.ф.-м.н., доцент
_____Т.А.Макаровских
_____2017 г.

Руководитель работы,
доцент каф. МиКМ,
к.т.н., доцент
_____В.И.Дударева
_____2017 г.

Автор работы
студент группы ЕТ-485
_____ Шумилов С.А.
«____»_____2017 г.

Челябинск, 2017

1 ОБЩИЕ СВЕДЕНИЯ

Пользовательский интерфейс предназначен для работы с облачным сервисом по автоматизации работы гидропонных установок. Используя интерфейс, пользователь может управлять активными итерациями.

2 ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Итерация посадки – период в который физически в гидропонные установки растет одна конкретная агрокультура. Итерация завершается по окончании выращивания урожая. Итерация считается активной до тех пор, пока она не завершена.

Ожидаемый срок созревания – ожидаемый, производителями семян, срок до созревания плодов.

3 ПРАВИЛА ВВОДА ВХОДНЫХ ДАННЫХ

Для создания новой итерации пользователю необходимо заполнить поля «Название посадки» и «Ожидаемый срок созревания» в форме создания новой итерации (рисунок 2).

- Название посадки – является текстовым полем. Максимальная длина поля – 100 символов;
- Ожидаемый срок созревания – числовое поле. В разработанной версии значение поля не может превышать 100.

4 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ И РАБОТА С ПРОГРАММОЙ

Пользователю предлагается интерфейс в виде веб страницы, на которой размещается шесть информационных блоков с графиками, блок с прогнозом срока выращивания (рисунок 1). Также при доступности создания итерации, на странице будет активен блок с созданием новой итерации (рисунок 2), а при наличие активной итерации, доступен блок с завершением данной итерации (рисунок 3).

Рисунок 1 – Внешний вид информационных блоков интерфейса

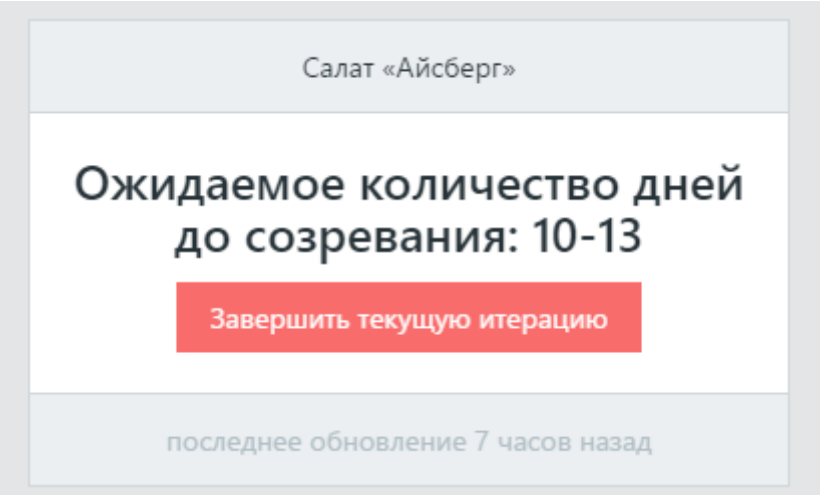

Рисунок 2 – Внешний вид интерфейса завершения итерации

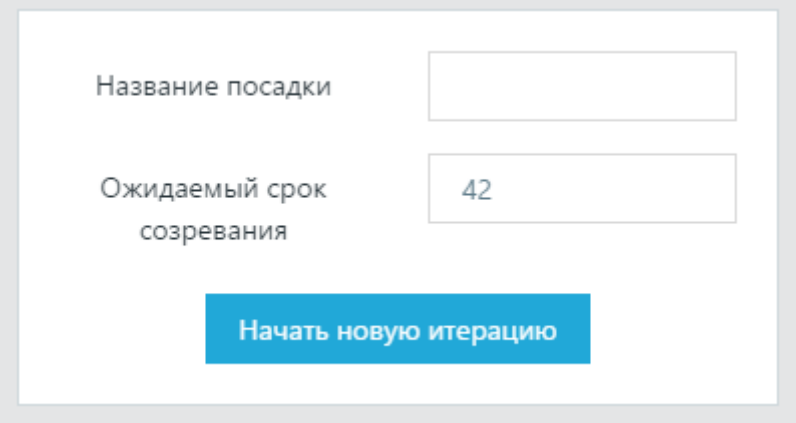

Рисунок 3 – Внешний вид интерфейса создания итерации

Каждый из 6 информационных блоков на рисунке 1 отображает одно из значений:

- уровень воды;
- уровень pH;
- уровень общего солесодержания;
- температура воды;
- температура воздуха;
- влажность воздуха.

5 СООБЩЕНИЯ ОБ ОШИБКАХ

При попытке создать итерацию с некорректными данными, такими как, длинное название посадки или неверное значение срока созревания, сервер сообщит клиентскому приложению о некорректности данных. После чего пользователю будет выведено информационное окно, сообщающее о некорректности введённых данных (рисунок 4).

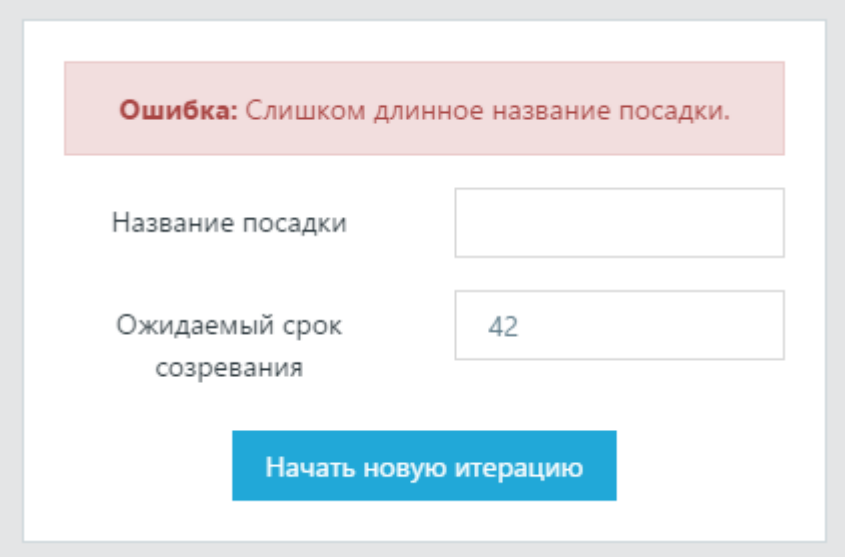

Рисунок 4 – Пример ошибки, возникающей при некорректности данных

ПРИЛОЖЕНИЕ В
ТЕКСТ ПРОГРАММЫ

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра «Математическое и компьютерное моделирование»

Анализ эффективности выращивания культурных растений
с использованием методов гидропоники и интернета вещей

ТЕКСТ ПРОГРММЫ
К ПРОГРАММНОМУ СРЕДСТВУ
ЮУрГУ – 01.03.02.2017.082.20.000 ТП ВКР

Нормоконтролер,
доцент каф. МиКМ,
к.ф.-м.н., доцент
_____Т.А.Макаровских
_____2017 г.

Руководитель работы,
доцент каф. МиКМ,
к.т.н., доцент
_____В.И.Дударева
_____2017 г.

Автор работы
студент группы ЕТ-485
_____ Шумилов С.А.
«____»_____2017 г.

Челябинск, 2017

Приложение В.1 – Файл `api.js`

```
var express = require('express');
var router = express.Router();

const transform = require("./transform.js");
const { statsRow, plants, commandsRow } = require('./db');
const { predictor } = require('./predict');
const { correction } = require('./corrections');

router.get('/addPlant', (req, res, next) => {
  if (!req.query.name) {
    return res.json({
      success: false,
      msg: 'send correct request',
    });
  }

  plants.findOne({ active: true }, (err, plant) => {
    if (err) {
      return res.json({
        success: false,
        msg: err,
      });
    }

    if (plant) {
      return res.json({
        success: false,
        msg: 'You already have active plant',
      });
    }

    const newPlant = new plants({
      name: req.query.name
    });

    newPlant.save((err, result) => {
      if (err) {
        return res.json({
          success: false,
          msg: err,
        });
      }

      return res.json({
        success: true,
      });
    });
  });
});

router.get('/completePlant', (req, res, next) => {
  if (!req.query.name) {
    return res.json({
      success: false,
      msg: 'send correct request',
    });
  }

  plants.findOne({ name: req.query.name }, (err, plant) => {
    if (err) {
      return res.json({
        success: false,
```

```
        msg: err,
      });
    }

    if (!plant) {
      return res.json({
        success: false,
        msg: 'You don\'t have any active plant',
      });
    }

    plant.active = false;

    plant.save((err, result) => {
      if (err) {
        return res.json({
          success: false,
          msg: err,
        });
      }

      res.json({
        success: true,
      });
    });
  });

});

router.get('/getStats', (req, res, next) => {
  plants.findOne({ active: true }, (err, plant) => {
    if (err) {
      return res.json({
        success: false,
        msg: err,
      });
    }

    statsRow
      .find({ name: plant.name })
      .limit(7)
      .exec((err, stats) => {
        if (err) {
          return res.json({
            success: false,
            msg: err,
          });
        }

        res.json({
          plant,
          stats,
        });
      });
  })
});

router.get('/addRow', (req, res, next) => {

  plants.findOne({ active: true }, (err, plant) => {
    if (err || !plant) {
      return res.json({
        success: false,
        msg: err,
```

```
      });
    }


    let row = new statsRow(Object.assign(
      {},
      req.query,
      {
        name: plant.name,
      }
    ));

    let command = correction(row, plant);
    let newCommandsRow = new commandsRow(commandsRow);

    newCommandsRow.save((err, result) => {
      if (err) {
        return res.json({
          success: false,
          msg: err,
        });
      }
    })

  // console.log(row); // 'Silence'

    row.save((err, result) => {
      if (err) {
        return res.json({
          success: false,
          msg: err,
        });
      }

      statsRow.find({ createdAt: {
        "$gte": plant.lastRecalculationDate,
        "$lt": new Date(),
      }}, (err, stats) => {
          console.log(stats);
        if (err || !stats || !stats.length || new Date(plant.lastRecalculationDate).getDate()
== new Date().getDate()) {
          return res.json({
            success: false,
            msg: err,
          });
        }

        let data = stats.reduce((prev, curr, ind) => {
          let tmp = {};
          if (curr.ph) {
            tmp.phSum = prev && prev.phSum ? prev.phSum + curr.ph : curr.ph;
            tmp.phCount = prev && prev.phCount ? prev.phCount + 1 : 1;
          }
          if (curr.ppm) {
            tmp.ppmSum = prev && prev.ppmSum ? prev.ppmSum + curr.ppm : curr.ppm;
            tmp.ppmCount = prev && prev.ppmCount ? prev.ppmCount + 1 : 1;
          }
          if (curr.temperature) {
            tmp.temperatureSum = prev && prev.temperatureSum ? prev.temperatureSum +
curr.temperature : curr.temperature;
            tmp.temperatureCount = prev && prev.temperatureCount ? prev.temperatureCount + 1 :
1;
          }
          if (curr.waterTemperature) {
```

```
            tmp.waterTemperatureSum = prev && prev.waterTemperatureSum ?
prev.waterTemperatureSum + curr.waterTemperature : curr.waterTemperature;
            tmp.waterTemperatureCount = prev && prev.waterTemperatureCount ?
prev.waterTemperatureCount + 1 : 1;
          }

          return tmp;
        }, {});
        data.ph = transform.normalizePh(data.phSum / data.phCount);
        data.ppm = transform.normalizeEC(data.ppmSum / data.ppmCount);
        data.temperature = transform.normalizeTemperature(data.temperatureSum /
data.temperatureCount);
        data.waterTemperature = transform.normalizeTemperature(data.waterTemperatureSum /
data.waterTemperatureCount);
        const numOfDay = ((new Date() - new Date(plant.createdAt)) /
(60*60*24*1000)).toFixed() / 100 + 0.01;

        plant.calculatedDays = predictor([
          , // № of day
          data.ph, // pH
          data.ppm, // EC
          data.waterTemperature, // waterTemperature
          data.temperature, // temperature
          plant.calculatedDays ? plant.calculatedDays : plant.recomendedDays / 100 // prevData
        ])[0];

        plant.lastRecalculationDate = new Date();

        plant.save((err, result) => {
          if (err) {
            return res.json({
              success: false,
              msg: err,
            });
          }

          res.json(command);
        });

      });
    });

  });
});

module.exports = router;
```

Приложение В.2 – Файл `correction.js`

```
exports.correction = (stats, plant) => {
  return {
    water: ((plant.waterCapacity - stats.water) / (plant.waterCapacity / (60 * 60 *
1000))).toFixed(),
    ppm: (plant.recomendedPpm - stats.ppm > 100) ? 500: 0,
    'ph+': (plant.recomendedPh - stats.ph > 1) ? 300: 0,
    'ph-': (plant.recomendedPh - stats.ph < 1) ? 300: 0,
    dayStart: Number,
    dayLength: Number,
    fanPower: Number,
  }
};
```

Приложение В.3 – Файл `db.js`

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/ferm');

var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', () => {});

const plantsSchema = mongoose.Schema({
  active: {
    type: Boolean,
    default: true,
  },
  recomendedDays: {
    type: Number,
    default: 65,
  },
  name: {
    type: String,
  },
  waterCapacity: {
    type: Number,
    default: 90,
  },
  recomendedPh: {
    type: Number,
    default: 5.5,
  },
  recomendedPpm: {
    type: Number,
    default: 1600,
  },
  recomendedTemperature: {
    type: Number,
    default: 26,
  },
  recomendedWaterTemperature: {
    type: Number,
    default: 22,
  },
  lastRecalculationDate: {
    type: Date,
    default: new Date(),
  },
  calculatedDays: Number,
}, {
  timestamps: true
});

const statsRowSchema = mongoose.Schema({
  name: String,
  water: Number,
  ph: Number,
  ppm: Number,
  temperature: Number,
  waterTemperature: Number,
  humidity: Number,
}, {
  timestamps: true
});
```

```
const commandsRowSchema = mongoose.Schema({
  water: Number,
  'ph+': Number,
  'ph-': Number,
  ppm: Number,
  dayStart: Number,
  dayLength: Number,
  fanPower: Number,
}, {
  timestamps: true
});

exports.plants = mongoose.model('plants', plantsSchema);
exports.statsRow = mongoose.model('statsRow', statsRowSchema);
exports.commandsRow = mongoose.model('commandsRow', commandsRowSchema);
```

Приложение В.4 – Файл `index.js`

```
var express = require('express');
var router = express.Router();

let { statsRow, plants } = require('./db');

/* GET home page. */
router.get('/', function(req, res, next) {

  plants.findOne({ active: true }, (err, plant) => {
    if (err || !plant) {
      return res.json({
        success: false,
        msg: err,
      });
    }

    console.log((plant.calculatedDays * 100).toFixed());
    statsRow.find({ name: plant.name }, (err, stats) => {
      console.log(stats);
    });
  });

  res.render('index', { title: 'Express' });
});

module.exports = router;
```

Приложение В.5 – Файл `learn.js`

```
var fs = require('fs');
var exported = require("./exported.json");
var transform = require("./transform.js");
var synaptic = require('synaptic');

function Gauss() {
  var ready = false;
  var second = 0.0;

  this.next = function(mean, dev) {
```

```
    mean = mean == undefined ? 0.0 : mean;
    dev = dev == undefined ? 1.0 : dev;

    if (this.ready) {
      this.ready = false;
      return this.second * dev + mean;
    } else {
      var u, v, s;
      do {
        u = 2.0 * Math.random() - 1.0;
        v = 2.0 * Math.random() - 1.0;
        s = u * u + v * v;
      } while (s > 1.0 || s == 0.0);

      var r = Math.sqrt(-2.0 * Math.log(s) / s);
      this.second = r * u;
      this.ready = true;
      return r * v * dev + mean;
    }
  };
}

norm = new Gauss();

var Neuron = synaptic.Neuron,
    Layer = synaptic.Layer,
    Network = synaptic.Network,
    Trainer = synaptic.Trainer,
    Architect = synaptic.Architect;

var perc = new Architect.Perceptron(6, 50, 1);

function getData(i, days, j) {
  return [
    i/100,
    transform.normalizePh(norm.next(6, 0.5)),
    transform.normalizeEC(norm.next(1600, 200)),
    transform.normalizeTemperature(norm.next(23, 2)),
    transform.normalizeTemperature(norm.next(25, 3)),
    days,
  ];
}

var imported = Network.fromJSON(exported);

var standalone = imported.standalone();


for (j = 0 ; j < 2000001; j+=1) {
  let defaultDays = Math.floor(Math.random() * 20) + 50;
  let prevDays = defaultDays / 100;
  var dynamicRate =  .035;
  for (i = 0 ; i < defaultDays; i+=1) {
    prevDays = perc.activate(getData(i, prevDays, j))[0];
    const days = norm.next(defaultDays - i, 1);

    perc.propagate(dynamicRate, [days / 100]);
    if (!(j % 10000))
      console.log(prevDays*100, days);
  }
  if (!(j % 10000))
    console.log('\n' + j + '---------------\n')
}
```

```
var json = JSON.stringify(perc.toJSON());

fs.writeFile('myjsonfile.json', json, 'utf8', () => {});
```

Приложение В.6 – Файл `transform.js`

```
// PH
exports.decodePh = (value) => {
    return -value / 828.24 * 14 + 7;
};

exports.encodePh = (value) => {
    return -(value * 59.16 - 414.14);
};

exports.normalizePh = (value) => {
    return value / 14;
};

exports.denormalizePh = (value) => {
    return value * 14;
};

// EC
exports.normalizeEC = (value) => {
    return value / 3000;
};

exports.denormalizeEC = (value) => {
    return value * 3000;
};

// Temperature
exports.normalizeTemperature = (value) => {
    return value / 40;
};

exports.denormalizeTemperature = (value) => {
    return value * 40;
};
```

Приложение В.7 – Файл `ferma.ino`

```
#include <ArduinoJson.h>
#include <EtherCard.h>
#include <Wire.h>
#include <OneWire.h>
#include <dht11.h>
#include <iarduino_RTC.h>


#define phSensor A0
#define ecSensor A1
#define waterTemperatureSensor 2
#define waterLevel A2
#define temperatureSensor A3
#define relay 4
#define pumpOne 3
#define pumpTwo 5
#define pumpThree 6
```

```
#define pumpFour 9

OneWire ds(ecSensor);
dht11 sensor;
iarduino_RTC time(RTC_DS1307);
const char server[] PROGMEM = "192.168.0.1";

struct ferm {
  int ph;
  int ppm;
  int waterTemperature;
  int waterLevel;
  int temperature;
  int humidity;
} ferm;

#define BUFFER_SIZE 500
byte Ethernet::buffer[500];
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };

void setup() {
  Serial.begin(9600);

  Wire.begin(1);
  Wire.onReceive(i2cEvent);

  if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
    Serial.println(F("Failed to access Ethernet controller"));
  Serial.println(F("Setting up DHCP"));
  if (!ether.dhcpSetup())
    Serial.println(F("DHCP failed"));
  ether.printIp("My IP: ", ether.myip);
  ether.printIp("Netmask: ", ether.netmask);
  ether.printIp("GW IP: ", ether.gwip);
  ether.printIp("DNS IP: ", ether.dnsip);

  time.begin();
  time.settime(0,0,0,02,06,17,5);

  TempProcess(0);
  controlFarm();
}

void loop() {
  uint16_t payloadPos = ether.packetLoop(ether.packetReceive());
  if (payloadPos) {
    readPayload(payloadPos);
  }
  if(millis()%1000 == 21600000) {
    sendData();
  }
}

void sendData() {
  String line = readSensors();
  char* req = "";
  for (int ii = 0; ii < line.length(); ++ii) {
    req += char(line[ii]);
  }
  ether.browseUrl(PSTR("/api/"), req, server, callback);
}

String readSensors() {
    ferm.waterLevel = getWaterLevel();
```

```
    String line = "addRow?water=" + ferm.waterLevel;
    ferm.ph = getPHLevel();
    line += "&ph=" + ferm.ph;
    ferm.temperature = getTemperature();
    line += "&temperature=" + ferm.temperature;
    ferm.temperature = getTemperature();
    line += "&waterTemperature=" + ferm.temperature;
    ferm.humidity = getHumidity();
    line += "&humidity=" + ferm.humidity;
    return line;
}

static void callback (byte status, word off, word len) {
}

void readPayload(uint16_t payloadPos) {
  String incomingData = (char *) Ethernet::buffer + payloadPos;
  Serial.println(incomingData);
  DynamicJsonBuffer jsonBuffer(500);
  JsonObject& data = jsonBuffer.parseObject(incomingData);
  setFarmSettings(data);
}

void setFarmSettings(JsonObject& data) {
  ferm.ph = data["ph"];
  ferm.waterLevel = data["water"];
  ferm.ppm = data["ppm"];
}

void controlFarm() {
  if (ferm.waterLevel > getWaterLevel()) {
    digitalWrite(pumpOne, HIGH);
  }
}

int getPHLevel() {;
  return analogRead(phSensor);
}

int getPPMLevel() {
  return analogRead(phSensor);
}

int getWaterTemperature() {
 return TempProcess(waterTemperatureSensor);
}

int getEcLevel() {
 return analogRead(ecSensor);
}

int getTemperature () {
  int chk = sensor.read(temperatureSensor);
  return sensor.temperature;
}

int getHumidity() {
  int chk = sensor.read(temperatureSensor);
  return sensor.humidity;
}

bool getWaterLevel() {
  return analogRead(waterLevel);
}
```

```
void i2cEvent(int) {

}


float TempProcess(bool ch) {
  //returns the temperature from one DS18B20 in DEG Celsius
  static byte data[12];
  static byte addr[8];
  static float TemperatureSum;
  if(!ch) {
    if ( !ds.search(addr)) {
      Serial.println("no more sensors on chain, reset search!");
      ds.reset_search();
      return 0;
    }
    if ( OneWire::crc8( addr, 7) != addr[7]) {
      Serial.println("CRC is not valid!");
      return 0;
    }
    if ( addr[0] != 0x10 && addr[0] != 0x28) {
      Serial.print("Device is not recognized!");
      return 0;
    }
    ds.reset();
    ds.select(addr);
    ds.write(0x44,1); // start conversion, with parasite power on at the end
  }
  else {
    byte present = ds.reset();
    ds.select(addr);
    ds.write(0xBE); // Read Scratchpad
    for (int i = 0; i < 9; i++) { // we need 9 bytes
      data[i] = ds.read();
    }
    ds.reset_search();
    byte MSB = data[1];
    byte LSB = data[0];
    float tempRead = ((MSB << 8) | LSB); //using two's compliment
    TemperatureSum = tempRead / 16;
  }
  return TemperatureSum;
}
```

Приложение В.8 – Файл `index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>Пользовательский интерфейс</title>
  <link href="css/style.css" rel="stylesheet">
</head>
<style>
  body {
  margin-top: 10px;
  }

  .fa {
```

```
  font-size: 1.5em;
  }
</style>
<body class="app aside-menu-fixed aside-menu-hidden">
  <div class="app-body">
    <main class="main">
      <div class="container-fluid">
        <div class="animated fadeIn">
          <div class="row">
            <div class="col-sm-6 col-lg-4">
              <div class="card card-inverse card-primary">
                <div class="card-block pb-0">
                  <button type="button" class="btn btn-transparent active p-0 float-right">
                  <i class="fa fa-tint" aria-hidden="true"></i>
                  </button>
                  <h4 class="mb-0">86.45</h4>
                  <p>Water level</p>
                </div>
                <div class="chart-wrapper px-3" style="height:150px;">
                  <canvas id="card-chart1" class="chart" height="150"></canvas>
                </div>
              </div>
            </div>
            <div class="col-sm-6 col-lg-4">
              <div class="card card-inverse card-danger">
                <div class="card-block pb-0">
                  <button type="button" class="btn btn-transparent active p-0 float-right">
                    <i class="fa fa-recycle" aria-hidden="true"></i>
                  </button>
                  <h4 class="mb-0">5.83</h4>
                  <p>pH level</p>
                </div>
                <div class="chart-wrapper px-3" style="height:150px;">
                  <canvas id="card-chart2" class="chart" height="150"></canvas>
                </div>
              </div>
            </div>
            <div class="col-sm-6 col-lg-4">
              <div class="card card-inverse card-warning">
                <div class="card-block pb-0">
                  <button type="button" class="btn btn-transparent active p-0 float-right">
                    <i class="fa fa-ioxhost" aria-hidden="true"></i>
                  </button>
                  <h4 class="mb-0">1678.83</h4>
                  <p>Ppm level</p>
                </div>
                <div class="chart-wrapper" style="height:150px;">
                  <canvas id="card-chart3" class="chart" height="150"></canvas>
                </div>
              </div>
            </div>
            <div class="col-sm-6 col-lg-4">
              <div class="card card-inverse card-danger">
                <div class="card-block pb-0">
                  <button type="button" class="btn btn-transparent active p-0 float-right">
                    <i class="fa fa-thermometer" aria-hidden="true"></i>
                  </button>
                  <h4 class="mb-0">28.5°C</h4>
                  <p>Water temperatures</p>
                </div>
                <div class="chart-wrapper px-3" style="height:150px;">
                  <canvas id="card-chart4" class="chart" height="150"></canvas>
                </div>
              </div>
```

```
            </div>
            <div class="col-sm-6 col-lg-4">
              <div class="card card-inverse card-info">
                <div class="card-block pb-0">
                  <button type="button" class="btn btn-transparent active p-0 float-right">
                    <i class="fa fa-thermometer" aria-hidden="true"></i>
                  </button>
                  <h4 class="mb-0">23.7°C</h4>
                  <p>Temperatures</p>
                </div>
                <div class="chart-wrapper px-3" style="height:150px;">
                  <canvas id="card-chart5" class="chart" height="150"></canvas>
                </div>
              </div>
            </div>
            <div class="col-sm-6 col-lg-4">
              <div class="card card-inverse card-success">
                <div class="card-block pb-0">
                  <button type="button" class="btn btn-transparent active p-0 float-right">
                    <i class="fa fa-percent" aria-hidden="true"></i>
                  </button>
                  <h4 class="mb-0">68.87%</h4>
                  <p>Humidity</p>
                </div>
                <div class="chart-wrapper px-3" style="height:150px;">
                  <canvas id="card-chart6" class="chart" height="150"></canvas>
                </div>
              </div>
            </div>
          </div>
          <div class="card text-center">
            <div class="card-header">
              Салат «Айсберг»
            </div>
            <div class="card-block">
              <h4 class="card-title">Ожидаемое количество дней до созревания: <span
id="prediction">10-13</span></h4>
              <button type="button" class="btn btn-danger">Завершить текущую итерацию</button>
            </div>
            <div class="card-footer text-muted">
              последнее обновление 7 часов назад
            </div>
          </div>
          <div class="card text-center">
            <div class="card-block">
              <div class="alert alert-danger" role="alert">
                <strong>Ошибка:</strong> Слишком длинное название посадки.
              </div>
              <div class="form-group row">
                <div class="col-4"></div>
                <label for="example-text-input" class="col-2 col-form-label">Название
посадки</label>
                <div class="col-1">
                  <input class="form-control" type="text" value="" id="example-text-input">
                </div>
              </div>
              <div class="form-group row">
                <div class="col-4"></div>
                <label for="example-number-input" class="col-2 col-form-label">Ожидаемый срок
созревания</label>
                <div class="col-1">
                  <input class="form-control" type="number" value="42" id="example-number-
input">
                </div>
```

```
              </div>
              <button type="button" class="btn btn-primary">Начать новую итерацию</button>
            </div>
          </div>
  <script src="bower_components/jquery/dist/jquery.min.js"></script>
  <script src="bower_components/tether/dist/js/tether.min.js"></script>
  <script src="bower_components/bootstrap/dist/js/bootstrap.min.js"></script>
  <script src="bower_components/pace/pace.min.js"></script>
  <script src="bower_components/chart.js/dist/Chart.min.js"></script>
  <script src="js/app.js"></script>
  <script src="js/views/main.js"></script>
</body>
</html>
```