

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

Направление подготовки 09.03.01 «Информатика и вычислительная техника»

Профиль «Программное обеспечение средств ВТ и
автоматизированных систем»

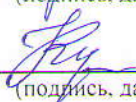
**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА БАКАЛАВРА**

на тему

Клиент-серверное приложение Погода

Студент  Акkuratнов Иван Михайлович
(подпись, дата)

Руководитель  Дубравин А.В.
(подпись, дата)

Нормоконтролёр  Кучин А.В.
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 03.06.2020 г. № 12)

Заведующий кафедрой  Митрохин М.А.
(подпись)

Работа защищена с отметкой *хорошо* (протокол заседания ГЭК от 22.06.20 № 13)

Секретарь ГЭК  Гурин Е.И.
(подпись)

Пенза, 2020

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

Вычислительная техника

«Утверждаю»
Заведующий кафедрой
п.п. М.А. Митрохин
30 апреля 2020 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Аккуратнов Иван Михайлович гр. 16BV2 факультета вычислительной техники.

2. Тема работы: «Клиент-серверное приложение погода».

Тема утверждена приказом ПГУ № 0335 а/о от "30" апреля 2020г.

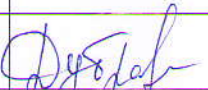
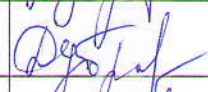
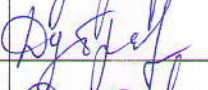
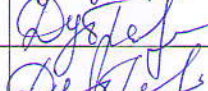
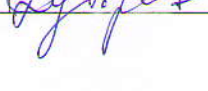
3. Руководитель бакалаврской работы к.т.н., доцент кафедры «ВТ» Дубравин А.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.).
Разработать приложения, сбора информации с датчика погода, температура, давление, влажность. Сохраняться данные датчика для ведения статистики. Разработать механизм просмотра данных погоды за определённый день. Разработка Rest API и страницы браузера для клиента.

5. Перечень подлежащих разработке вопросов.

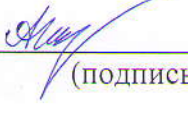
Требуется произвести анализ предметных областей: изучить типы СУБД для хранения данных, виды архитектуры для веб-приложений и рассмотреть основные языки для веб-разработки. Необходимо изучить методологии проектирования веб-приложений, а также изучить функционал фреймворка Spring, изучить аппаратную платформу Arduino и совместимые с ней датчики погоды.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	10	30.04.2020 – 03.05.2020	
2.	Анализ требований	10	03.05.2020 – 05.05.2020	
3.	Изучение технических средств решения поставленной задачи	15	05.05.2020 – 08.05.2020	
4.	Разработка и тестирование приложения	45	08.05.2020 – 31.05.2020	
5.	Оформление пояснительной записки	10		

Дата выдачи задания "30" апреля 2020 г.

Руководитель бакалаврской работы  Дубравин А.В.
(подпись, дата)

Задание к исполнению принял студент  Акkuratнов И.М.
(подпись, дата)

Работу к защите допустить
Декан факультета  Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа содержит 70 листов, 9 рисунка, 7 листингов, 9 источников, 2 приложения.

КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ, БД, СУБД, REST, API, JAVA, SPRING, ARDUINO, VUE. VUETIFY.

Цель работы – разработать клиент-серверное приложение «Погода».

Объектом разработки является приложение, обеспечивающее выполнение следующих функций: сбор, хранение информации о погоде. Создание API для получения информации на странице браузера.

В результате проведенной работы было разработано приложение, удовлетворяющее поставленной задаче, а именно: разработана структура БД, создан *REST API*, создана страница браузера, создана система сбора информации. Приложение было протестировано и успешно отлажено.

					ПГУ 2.090301.01.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата	<i>Клиент-серверное приложение Погода Пояснительная записка</i>		
Разраб.		<i>Аккуратнов И.М.</i>					
Провер.		<i>Дубравин А.В.</i>					
Н. Контр		<i>Кучин А.В.</i>					
Утверд.		<i>Митрохин А.В.</i>			ФВТ, каф. ВТ <i>гр. 16ВВ2</i>		
					Лит.	Лист	Листов
						2	120

Содержание

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ.....	5
ВВЕДЕНИЕ.....	6
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	7
1 Анализ предметной области.....	7
1.1 Анализ СУБД	7
1.2 Выбор аппаратной платформы	13
1.3 Выбор языка программирования	15
1.4 Описание архитектуры REST API.....	17
1.5 Выбор фреймворка для Frontend разработки	19
1.6 Выбор системы управления версиями зависимостей для Java	24
1.7 Выбор пакетного менеджера для JavaScript.....	27
1.8 Описание серверного механизма Thymeleaf.	28
1.9 Описание фреймворка Spring.....	30
1.10 Выводы.....	31
2 РЕАЛИЗАЦИЯ.....	33
2.1 Подключение датчика к плате	33
2.2 Разработка прошивки датчика погоды	34
2.3 Проектирование серверной части	35
2.4 Проектирование базы данных разрабатываемой системы	36
2.5 Разработка Rest API	37
2.6 Разработка Web-страницы.....	38
2.7 Взаимодействие классов.....	38
3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ	41
3.1 Результат работы приложения со стороны пользователя системы	41
3.2 Выводы.....	45
4 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ	46

Изм.	Лист	№ докум.	Подпись	Дата

ЗАКЛЮЧЕНИЕ	48
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	49
ПРИЛОЖЕНИЯ.....	50
Приложение А Листинги программы на JavaScript.....	50
Приложение В Презентация	67

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API (англ. Application Programming Interface) – программный интерфейс приложения;

HTML (англ. HyperText Markup Language) – язык гипертекстовой разметки;

HTTP (англ. HyperText Transfer Protocol) – протокол передачи гипертекста;

SSR (англ. Server Side Rendering) – рендеринг на серверной стороне;

CSR (англ. Client Side Rendering) – рендеринг на клиентской стороне;

SSL (англ. Secure Sockets Layer) – уровень защищённых сокетов;

TLS (англ. Transport Layer Security) – протокол защиты транспортного уровня;

TTI (англ. Time To Interactive) – время до интерактивности;

FCP (англ. First Contentful Paint) – первая содержательная отрисовка;

MVVM (англ. Modal-View-ViewModal) - шаблон проектирования архитектуры приложения;

MVC (англ. Modal-View-Controller) - шаблон проектирования архитектуры приложения;

DOM (англ. Document Object Model) - объектная модель документа;

UI (англ. User Interface) – пользовательский интерфейс;

NPM (англ. Node Package Manager) – пакетный менеджер Node.js;

UML (англ. Unified Modeling Language) – унифицированный язык моделирования.

ВВЕДЕНИЕ

Отслеживание погодных условий и поддержание необходимых показателей для огромного количества отраслей производства просто необходимо. Ни одна даже самая хорошо построенная теплица не сможет выполнять свою основную функцию, выращивание растений, без правильного температурного режима. Поддержание комфортной температуры в квартире с использованием обогревающих технологий невозможно без датчиков температуры. Поддержание прохладной температуры необходимо при хранении продуктов.

В данной работе мы разберем возможность мониторинга погодных условий. Создание базы данных для сбора показателей за определенное время. Создадим доступ для комфортного просмотра этих данных.

					ПГУ 2.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

При разработке программных продуктов необходимо провести анализ существующих вариантов для решения задачи. В данном разделе рассматриваются общие теоретические вопросы, освещающие технологические аспекты, необходимые для реализации программного продукта корпоративного класса. В частности, производится выбор архитектуры приложения, языка программирования и архитектуры. В каждом подразделе обосновывается выбор того или иного технического решения задачи.

1 Анализ предметной области

При разработке программных продуктов необходимо провести анализ существующих вариантов для решения задачи. В данном разделе рассматриваются общие теоретические вопросы, освещающие технологические аспекты, необходимые для реализации программного продукта. В частности, производится выбор СУБД, аппаратной архитектуры и датчика для сбора данных, архитектуры приложения, языка программирования. В каждом подразделе обосновывается выбор того или иного технического решения задачи.

1.1 Анализ СУБД

1.1.1 Oracle

Неудивительно, что корпорация Oracle предлагает одноимённый продукт, с которого обычно начинается рассмотрение вариантов популярных СУБД. Первая версия Oracle была создана в конце 70-х годов, имея на данный момент блестящую репутацию. Кроме того, существует несколько версий этого продукта для удовлетворения потребностей конкретной организации.

Актуальная версия Oracle на момент написания настоящей статьи - 19с - предназначена для облачных сред и может быть размещена на одном или нескольких серверах, это позволяет управлять базами данных, которые содержат миллиарды записей. Некоторые из функций новейшей версии Oracle включают в себя grid framework и использования как физических, так и логических структур.

Это означает, что физическое управление данными не влияет на доступ к логическим структурам. Кроме того, безопасность в этой версии доведена до высочайшего уровня, потому что каждая транзакция изолирована от других.

Достоинства

Самые свежие инновации и впечатляющий функционал уже внедрены в этом продукте, поскольку компания Oracle стремится держать планку даже на фоне других разработчиков СУБД.

СУБД от Oracle является крайне надёжной, фактически это эталон надёжности среди подобных систем.

Недостатки

Стоимость Oracle может оказаться непомерно высокой, особенно для небольших организаций.

Система может потребовать значительных ресурсов уже сразу после установки, поэтому возможно потребуется модернизировать оборудование для внедрения Oracle.

Идеально подходит для крупных организаций, которые работают с огромными базами данных и разнообразными функциями.

1.1.2 MySQL

MySQL - одна из самых популярных баз данных для веб-приложений. Фактически, является стандартом de facto для веб-серверов, которые работают под управлением операционной системы Linux. MySQL - это бесплатный пакет программ, однако новые версии выходят постоянно, расширяя функционал и улучшая безопасность. Существуют специальные платные версии, предназначенные для коммерческого использования. В бесплатной версии наибольший упор делается на скорость и надежность, а не на полноту функционала, который может стать и достоинством и недостатком - в зависимости от области внедрения.

Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

Эта СУБД позволяет выбирать различные движки для системы хранения, которые позволяют менять функционал инструмента и выполнять обработку данных, хранящихся в различных типах таблиц. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

Она также имеет простой в использовании интерфейс, и пакетные команды, которые позволяют удобно обрабатывать огромные объемы данных. Система невероятно надежна и не стремится подчинить себе все доступные аппаратные ресурсы.

Достоинства

- Распространяется бесплатно
- Прекрасно документирована
- Предлагает много функций, даже в бесплатной версии
- Пакет MySQL включен в стандартные репозитории наиболее распространенных дистрибутивов операционной системы Linux, что позволяет устанавливать её элементарно
- Поддерживает набор пользовательских интерфейсов

Недостатки

- Придётся потратить много времени и усилий, чтобы заставить MySQL выполнять несложные задачи, хотя другие системы делают это автоматически, например: создавать инкрементные резервные копии.
- Отсутствует встроенная поддержка XML или OLAP.
- Для бесплатной версии доступна только платная поддержка.
- Идеально подходит для организаций, которым требуется надежный инструмент управления базами данных, но бесплатный.

1.1.3 Microsoft SQL сервер

Ещё одной из популярных СУБД является программный продукт Microsoft SQL-сервер. Это система управления базами данных, движок которой работает на облачных серверах, а также локальных серверах, причем можно комбинировать типы применяемых серверов одновременно. Вскоре

после выпуска Microsoft SQL сервер 2016, Microsoft адаптировала продукт для операционной системы Linux, а на Windows-платформе он работал изначально.

Одной из уникальных особенностей версии 2016 года является temporal data support (временная поддержка данных), которая позволяет отслеживать изменения данных с течением времени. Последняя версия Microsoft SQL-сервер поддерживает dynamic data masking (динамическую маскировку данных), которая гарантирует, что только авторизованные пользователи будут видеть конфиденциальные данные.

Достоинства

- Продукт очень прост в использовании
- Текущая версия работает быстро и стабильно
- Движок предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов.
- Вы сможете получить доступ к визуализации на мобильных устройствах.
- Он очень хорошо взаимодействует с другими продуктами Microsoft.

Недостатки

- Цена для юридических лиц оказывается неприемлемой для большей части организаций.
- Даже при тщательной настройке производительности SQL Server способен занять все доступные ресурсы.
- Сообщается о проблемах с использованием службы интеграции для импорта файлов.

Идеально подходит для: крупных организаций, которые уже используют ряд продуктов Microsoft.

1.1.4 PostgreSQL

PostgreSQL является одним из нескольких бесплатных популярных вариантов СУБД, часто используется для ведения баз данных веб-сайтов. Это была одна из первых разработанных систем управления базами данных, поэтому в настоящее время она хорошо развита, и позволяет пользователям управлять как структурированными, так и неструктурированными данными. Может быть использован на большинстве основных платформ, включая Linux. Прекрасно справляется с задачами импорта информации из других типов баз данных с помощью собственного инструментария.

Движок БД может быть размещен в ряде сред, в том числе виртуальных, физических и облачных. Самая свежая версия, PostgreSQL 12, предлагает обработку больших объемов данных и увеличение числа одновременно работающих пользователей. Безопасность была улучшена благодаря поддержке DBMS_SESSION.

Достоинства

- Является масштабируемым и способен обрабатывать терабайты данных.
- Поддерживает формат json.
- Существует множество predefined функций.
- Доступен ряд интерфейсов.

Недостатки

- Документация туманна, поэтому, возможно, ответы на некоторые вопросы придется искать в интернете.

- Конфигурация может смутить неподготовленного пользователя.
- Скорость работы может падать во время проведения пакетных операций или выполнения запросов чтения.

Идеально подходит для организаций с ограниченным бюджетом, но квалифицированными специалистами, когда требуется возможность выбрать свой интерфейс и использовать json.

1.2 Выбор аппаратной платформы

1.2.1 Arduino UNO

Arduino Uno является стандартной платой Arduino и возможно наиболее распространенной. Она основана на чипе ATmega328, имеющем на борту 32 КБ флэш-памяти, 2 Кб SRAM и 1 Кбайт EEPROM памяти. На периферии имеет 14 дискретных (цифровых) каналов ввода / вывода и 6 аналоговых каналов ввода / вывода, это очень разносторонне-полезные девайсы, позволяющие покрывать большинство любительских задач в области микроконтроллерной техники. Данная плата контроллера является одной из самых дешевых и наиболее часто используемых. При планировании нового проекта, если вы незнакомы, с платформой Arduino, советуем начать с Uno.

1.2.2 Arduino MEGA

Как Arduino Uno, но на базе более мощного микроконтроллера той же архитектуры. Отличный выбор «на вырост» или если Arduino Uno перестала справляться. В разы больше памяти: 256 КБ против 32 КБ постоянной и 8 КБ против 2 КБ оперативной. В разы больше портов: 60 из них 16 аналоговых и 15 с ШИМ. Немного длиннее базовой Arduino Uno: 101×53 мм против 69×53 мм.

1.2.3 Arduino NANO

Это наш выбор. Arduino Nano — это функциональный аналог *Arduino Uno*, но размещённый на миниатюрной плате. Отличие заключается в отсутствии собственного гнезда для внешнего питания, использованием чипа FTDI FT232RL для USB-Serial преобразования (либо CH340G, требуется установить соответствующие драйвера) и применением mini-USB кабеля для взаимодействия вместо стандартного. В остальном, начинка и способы взаимодействия совпадают с базовой моделью. Платформа имеет штырьковые контакты, что позволяет легко устанавливать её на макетную плату. Используйте Arduino Nano там, где важна компактность, а возможностей Arduino Mini либо недостаточно, либо не хочется заниматься пайкой.

1.2.4 Датчик BME280

Это наш выбор. Данный датчик прост, предварительно откалиброван и для подключения не требуется дополнительных компонентов. Данный датчик измеряет влажность, температуру и давление с помощью данных показаний осуществляется расчет высоты.

Технические параметры

- Напряжение питания: 3.3 В – 5 В
- Рабочий ток: 1 мА
- Диапазон измерения давления: 300 гПа – 1100 гПа (точность ± 1.0 гПа)
- Диапазон измерения температуры: -40°C до $+85^{\circ}\text{C}$ (точность $\pm 0.5^{\circ}\text{C}$)
- Диапазон измерения влажности: 20 % до 80 % (точность ± 3 %)
- Интерфейс: I2C
- Габариты: 12 мм x 10 мм

1.3 Выбор языка программирования

При реализации любого программного продукта одним из самых важных этапов является выбор языка программирования. От этого зависит продуктивность и чистота кода, а также длительность периода создания минимально жизнеспособного продукта.

В настоящее время наиболее популярными языками программирования для разработки веб-приложений являются: *Go*, *Java*, *JavaScript*, *Python*. Такие языки как *C++*, *Php* и *Ruby* в последнее время теряют популярность в веб-разработке.

1.3.1 Язык программирования *Go*

Go представляет компилируемый статически типизированный язык программирования от компании Google. Язык *Go* предназначен для создания различного рода приложений, но прежде всего это веб-сервисы и клиент-серверные приложения. Хотя также язык обладает возможностями по работе с графикой, низкоуровневыми возможностями.

Но данный язык не является объекто-ориентированным в полном смысле этого понятия, так как в язык сознательно не была включена концепция наследования. Язык обладает меньшей переносимостью на разные платформы, так как компилируется в машинный код.

1.3.2 Язык программирования *Python*

Python – высокоуровневый язык программирования, который ориентирован на повышение производительности разработчика и читаемости кода. Этот язык очень гибкий и обладает широким функционалом, а код высокой степенью читаемости. Но у данного языка есть существенный недостаток – он обладает низким быстродействием и поэтому часто используется как язык второго плана.

1.3.3 Язык программирования JavaScript

JavaScript – мультипарадигменный язык программирования [4][10]. Язык не принадлежит какой-либо организации или компании. Наиболее часто используется в качестве языка сценариев веб-страниц. *JavaScript* поддерживает императивный, декларативный и объектно-ориентированный стили программирования. *JavaScript* часто используется в таких программных продуктах, которые в свою очередь используются в веб-разработке. Так как язык является интерпретируемым, то производительность и безопасность зависит от конкретной реализации интерпретатора («движка»).

1.3.4 Язык программирования Java

Java – сильно типизированный объектно-ориентированный язык программирования [11]. В настоящее время разработкой занимается компания *Oracle*. Основной особенностью данного языка является то, что программы на *Java* транслируются в байт-код, который выполняется на виртуальной машине *Java*. Виртуальная машина *Java* в свою очередь является интерпретатором и передает инструкции оборудованию, что обеспечивает легкую переносимость программы на разные платформы. В настоящее время *Java* является одним из самых популярных языков программирования.

1.3.5 Выбор языка программирования

В качестве языка программирования был выбран язык *Java* для *backend* разработки и *JavaScript* для *frontend* разработки. *Java* и *JavaScript* давно используется программистами, имеет большое сообщество, хорошую документацию, множество популярных фреймворков. Для упрощения и ускорения процесса разработки для *backend* был выбран фреймворк *Spring*. Данный фреймворк является универсальным и используется для создания бизнес-приложений.

1.4 Описание архитектуры REST API

REST (Representational state transfer) – это стиль архитектуры программного обеспечения для распределенных систем, таких как World Wide Web, который, как правило, используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами.

В общем случае REST является очень простым интерфейсом для взаимодействия компонентов распределённого приложения в сети. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат.

1.4.1 Использование REST для построения Web-сервисов

Как известно, web-сервис – это приложение, работающее в World Wide Web и доступ, к которому предоставляется по HTTP-протоколу. Обмен информацией идет с помощью форматов XML, JSON.

Для каждой единицы информации (info) определяется 5 действий. А именно:

GET /info/ (Index) – получает список всех объектов. Как правило, это упрощенный список, т.е. содержащий только поля идентификатора и названия объекта, без остальных данных.

GET /info/{id} (View) – получает полную информацию о объекте.

PUT /info/ или POST /info/ (Create) – создает новый объект. Данные передаются в теле запроса без применения кодирования, даже urlencode. В Java с использованием Spring тело запроса может быть получено с помощью аннотации @RequestBody:

@PostMapping

```

public int sendEmail (@RequestBody String message){
    return 1;
}

```

POST /info/{id} или PUT /info/{id} (Edit) – изменяет данные с идентификатором {id}, возможно заменяет их. Данные так же передаются в теле запроса, но в отличие от PUT здесь есть некоторый нюанс. Дело в том, что POST-запрос подразумевает наличие urldecoded-post-data. Т.е. если не применять кодирования – это нарушение стандарта. Тут кто как хочет – некоторые не обращают внимания на стандарт, некоторые используют какую-нибудь post-переменную.

DELETE /info/{id} (Delete) – удаляет данные с идентификатором {id}.

Еще раз отмечу, что в нашем примере /info/ — может и базироваться на какой-то другой информации, что может быть (и должно) быть отражено в URL:

/data/4/otherdata/6/info/3/ ... и тому подобное.

1.4.2 Практическое применение

Самое главное достоинство сервисов в том, что с ними работать может, какая угодно система, будь то сайт, flash, программа и др. так как методы парсинга XML и выполнения запросов HTTP присутствуют почти везде.

Архитектура REST позволяет серьезно упростить эту задачу. Конечно в реальности, того что описано недостаточно, ведь нельзя кому угодно давать возможность изменять информацию, то есть нужна еще авторизация и аутентификация. Но это достаточно просто разрешается при помощи различного типа сессий или просто HTTP Authentication.

1.5 Выбор фреймворка для Frontend разработки

Фронтендом называют клиентскую сторону пользовательского интерфейса по отношению к программно-аппаратной части сервиса.

Говоря простым языком, frontend — это разработка функциональности и пользовательского интерфейса, работающих на клиентской стороне приложения или веб-сайта. Сюда относят всё, что пользователь видит, открывая веб-страницу. Для создания удобного и востребованного продукта фронтенд-разработчики сотрудничают с программистами, дизайнерами, UX-аналитиками.

1.5.1 Фреймворк Angular

Angular — это среда JavaScript MVVM, основанная в 2009 году, которая отлично подходит для создания интерактивных веб-приложений.

Преимущества Angular:

- Angular используется вместе с Typescript. Он имеет исключительную поддержку для этого.
- Angular-language-service — обеспечивает интеллектуальные возможности и автозаполнение шаблона HTML-компонента.
- Новые функции, такие как generation Angular, использующие библиотеки npm из CLI, generation, и разработка компонентов, использующая Angular.
- Подробная документация, позволяющая разработчику получить всю необходимую информацию, не прибегая к помощи его коллег. Однако это требует больше времени для обучения.

Изм.	Лист	№ докум.	Подпись	Дата

- Односторонняя привязка данных, которая обеспечивает исключительное поведение приложения, что сводит к минимуму риск возможных ошибок.
- MVVM (Model-View-ViewModel), которая позволяет разработчикам работать отдельно над одним и тем же разделом приложения, используя один и тот же набор данных.
- Внедрение зависимостей от компонентов, связанных с модулями и модульностью в целом.
- Структура и архитектура, специально созданные для большой масштабируемости проекта.

Недостатки Angular:

- Разнообразие различных структур (Injectables, Components, Pipes, Modules и т. д.) усложняет изучение по сравнению с React и Vue.js, у которых есть только «Component».
- Относительно медленная производительность, учитывая различные показатели. С другой стороны, это можно легко решить, используя так называемый «ChangeDetectionStrategy», который помогает вручную контролировать процесс рендеринга компонентов.

Компании, использующие Angular: Microsoft, Autodesk, MacDonald's, UPS, Cisco Solution Partner Program, AT&T, Apple, Adobe, GoPro, ProtonMail, Clarity Design System, Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase.

1.5.2 Фреймворк React

					ПГУ 2.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

React — это библиотека JavaScript, разработанная Facebook в 2013 году, которая отлично подходит для создания современных одностраничных приложений любого размера и масштаба.

Преимущества React:

- Легко изучить, благодаря простому дизайну, использованию JSX (HTML-подобный синтаксис) для шаблонов и очень подробной документации. Разработчики тратят больше времени на написание современного JavaScript и меньше беспокоятся о коде, специфичном для фреймворка.
- Очень быстрая, благодаря реализации React Virtual DOM и различным оптимизациям рендеринга.
- Отличная поддержка рендеринга на стороне сервера, что делает его мощной платформой для контент-ориентированных приложений.
- Первоклассная поддержка Progressive Web App (PWA) благодаря генератору приложений `create-react-app`.
- Привязка данных является односторонней, что означает меньше нежелательных побочных эффектов.
- Redux, самая популярная платформа для управления состоянием приложений в React, ее легко учить и использовать.
- React реализует концепции функционального программирования (FP), создавая простой в тестировании и многократно используемый код.

- Приложения могут быть созданы с помощью TypeScript или Facebook's Flow, имеющими встроенную поддержку JSX.
- Переход между версиями, как правило, очень прост: Facebook предоставляет «кодовые модули» для автоматизации большей части процесса.
- Навыки, полученные в React, могут быть применены к разработке на React Native.

Недостатки React:

- React не однозначен и оставляет разработчикам возможность выбирать лучший способ развития. Это может быть решено сильным лидерством проекта и хорошими процессами.
- Сообщество делится по способам написания CSS в React, которые разделяются на традиционные таблицы стилей (CSS Modules) и CSS-in-JS (т.е. Emotion и Styled Components).
- React отходит от компонентов на основе классов, что может стать препятствием для разработчиков, которым более комфортно работать с объектно-ориентированным программированием (ООП).
- Смешивание шаблонов с логикой (JSX) может сбить с толку некоторых разработчиков при первых знакомствах с React.

Компании, использующие React: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft, Slack, Storybook и многие другие

1.5.3 Фреймворк Vue.js

Vue.js — это JavaScript-фреймворк, основанный в 2013 году, который идеально подходит для создания высоко адаптируемых пользовательских интерфейсов и сложных одностраничных приложений.

Преимущества Vue.js:

- Усиленный HTML. Это означает, что Vue.js имеет много характеристик схожих с Angular, а это, благодаря использованию различных компонентов, помогает оптимизации HTML- блоков.
- Подробная документация. Vue.js имеет очень подробную документацию, которая может ускорить процесс обучения для разработчиков и сэкономить много времени на разработку приложения, используя только базовые знания HTML и JavaScript.
- Адаптивность. Может быть осуществлен быстрый переход от других фреймворков к Vue.js из-за сходства с Angular и React с точки зрения дизайна и архитектуры.
- Потрясающая интеграция. Vue.js можно использовать как для создания одностраничных приложений, так и для более сложных веб-интерфейсов приложений. Важно, что небольшие интерактивные элементы можно легко интегрировать в существующую инфраструктуру без негативных последствий.
- Масштабирование. Vue.js может помочь в разработке довольно больших шаблонов многократного использования, которые могут быть сделаны почти за тоже время, что и более простые.

- Крошечный размер. Vue.js весит около 20 КБ, сохраняя при этом свою скорость и гибкость, что позволяет достичь гораздо лучшей производительности по сравнению с другими платформами.

Недостатки Vue.js:

- Недостаток ресурсов. Vue.js по-прежнему занимает довольно небольшую долю рынка по сравнению с React или Angular, что означает, что обмен знаниями в этой среде все еще находится на начальной стадии.
- Риск чрезмерной гибкости. Иногда у Vue.js могут возникнуть проблемы при интеграции в огромные проекты, и пока еще нет опыта возможных решений, но они обязательно появятся в ближайшее время.

Компании, которые используют Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab и Laracasts, Adobe, Behance, Codeship, Reuters.

1.6 Выбор системы управления версиями зависимостей для Java

Главное отличие между Maven и Gradle является то, что Maven - это инструмент управления и понимания проектов программного обеспечения, который управляет сборками проектов, отчетами и документами, а Gradle - это инструмент автоматизации сборки с открытым исходным кодом, ориентированный на гибкость и производительность

1.6.1 Maven

Maven - это инструмент управления проектами, который управляет сборками проектов, зависимостями, распространением, выпусками и т. Д. Здесь сборка программного обеспечения относится к процессу, с помощью которого исходный код преобразуется в автономную форму, которую можно запустить на компьютере.

Кроме того, Maven позволяет компилировать, распространять, документировать и совместно работать в команде. Кроме того, репозиторий Maven - это каталог упакованного файла JAR с файлом pom.xml. Он имеет информацию о конфигурации для создания проекта. Здесь JAR - это пакет, который объединяет несколько файлов классов Java и связанных ресурсов в один файл для распространения.

1.6.2 Gradle

Gradle - это система автоматизации сборки с открытым исходным кодом. Он использует понятия Apache Ant и Apache Maven. Прежде всего, он был разработан для поддержки многопроектных сборок. Где, Gradle использует направленный ациклический график, чтобы определить порядок выполнения задач. Gradle стал популярным в течение короткого периода времени. Например, Google принял Gradle в качестве инструмента сборки по умолчанию для ОС Android.

Кроме того, Gradle основан на доменно-специфическом языке (DSL). Он не использует XML. Потому что он предназначен для решения конкретной проблемы. Кроме того, он поддерживает жизненный цикл программного обеспечения от компиляции до статистического анализа и тестирования до упаковки и развертывания.

Кроме того, Gradle предоставляет множество преимуществ. Это позволяет применять общие принципы проектирования к процессу сборки. Это помогает разработчикам создавать хорошо структурированные и поддерживаемые сборки для нескольких проектов. Кроме того, он предоставляет различные методы для управления зависимостями.

1.6.3 Сравнение рассматриваемых систем управления зависимостями

Скрипты сборки Gradle, написанные на Groovy, на порядок короче XML, используемых Maven. Разница заметна уже на hello world.

Hello world на Maven:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>edu.maven</groupId>
  <artifactId>hello-maven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Hello world на Gradle:

```
// build.gradle
apply plugin: 'java'
repositories {
    mavenCentral()
}
dependencies {
    compile 'org.slf4j:slf4j-api:1.7.25'
    testCompile 'junit:junit:4.12'
}
```

Безусловно, Gradle быстрее Maven. Чтобы Gradle максимально быстро собирал ваш проект, в нём используются всевозможные кэши и даже специальный процесс – Gradle Daemon, живущий после сборки. Но если вы отключите Gradle Daemon на CI (или осуществляете сборку не так часто), то прирост в скорости станет не такой очевидный.

1.7 Выбор пакетного менеджера для JavaScript

Npm и Yarn – два самых популярных менеджеры пакетов для JavaScript. Менеджеры пакетов являются способом автоматизации процесса установки, обновления и удаления сторонних модулей, которые хранятся в общей базе модулей. В этой статье будет рассказано в чем разница между двумя самыми популярными менеджерами пакетов для JavaScript – npm и Yarn.

Npm – это менеджер пакетов, входящий в состав Node.js. Он использует клиент командной строки и базу данных, состоящую из общедоступных и частных пакетов, известной как npm registry. Пользователи могут получить доступ к базе через сайт или через консоль.

Yarn был разработан в Facebook чтобы избавиться от недостатков npm. Технически Yarn не является заменой npm, так он берет информацию про модули из базы npm. По сути Yarn это новый установщик, который по-прежнему базируется на структуре, заданной npm. В Yarn доступны все те же пакеты что и в npm, поэтому, переезд с npm на Yarn не требует больших усилий. Yarn - это независимый проект с открытым исходным кодом, не привязанный ни к одной компании.

Быстро: пряжа кэширует каждый загруженный пакет, поэтому ему больше не нужно загружать тот же пакет. Он также делает почти все одновременно, чтобы максимизировать использование ресурсов. Это означает, что установка происходит еще быстрее.

Надежность: используя подробный, но лаконичный формат файла блокировки и детерминированный алгоритм для операций установки, Yarn может гарантировать, что любая установка, работающая в одной системе, будет работать точно так же в другой системе.

Безопасный: Yarn использует контрольные суммы для проверки целостности каждого установленного пакета перед выполнением его кода.

По этим 3 критериям: быстрота, надежность и безопасность, я и выбирал yarn.

1.8 Описание серверного механизма Thymeleaf.

Thymeleaf — современный серверный механизм Java-шаблонов для веб- и автономных сред, способный обрабатывать HTML, XML, JavaScript, CSS и даже простой текст.

Основной целью Thymeleaf является создание элегантного и удобного способа шаблонизации. Чтобы достичь этого, Thymeleaf основывается на концепции Natural Templates, чтобы внедрить свою логику в файлы шаблонов таким образом, чтобы этот шаблон не влиял на отображения прототипа дизайна.

Thymeleaf также был разработан с самого начала с учетом стандартов Web, особенно HTML5, что позволяет вам создавать полностью соответствующие стандарту шаблоны.

Из коробки, Thymeleaf позволяет работать с шестью типами шаблонов, каждый из которых называется «Формат шаблона/Template Mode»:

- HTML
- XML

- TEXT
- JAVASCRIPT
- CSS
- RAW

Итого: два markup формата (HTML and XML), три текстовых формата (TEXT, JAVASCRIPT и CSS) и простой формат (RAW).

HTML формат подразумевает HTML стандарт, включая HTML5, HTML 4 и XHTML. Валидация не выполняется, и код/структура оригинального шаблона будут соблюдаться в максимально возможной степени.

Формат шаблона XML позволит создать XML шаблон. В этом случае ожидается, что код будет хорошо сформирован, иначе без закрытых тегов, без кавычек атрибутов и прочего парсер сгенерирует исключения. Обратите внимание, что не будет проведена проверка DTD или XML-схемы.

Режим шаблона TEXT позволяет использовать специальный синтаксис для шаблонов без разметки. Примерами таких шаблонов могут быть текстовые письма или шаблонные документы. Обратите внимание, что HTML или XML-шаблоны также могут обрабатываться как TEXT, и в этом случае разметка не анализируется, и каждый тег, DOCTYPE, комментарий и прочее будет рассматриваться как простой текст.

Режим шаблона JAVASCRIPT позволит обрабатывать файлы JavaScript в приложении Thymeleaf. Это означает возможность использовать данные моделей внутри файлов JavaScript так же, как это можно сделать в HTML-файлах, но с интеграциями, специфичными для JavaScript, такими как

специализированное экранирование или скриптинг. Режим шаблона JAVASCRIPT считается текстовым режимом и поэтому использует тот же специальный синтаксис, что и режим шаблона TEXT.

Режим шаблона CSS позволяет обрабатывать файлы CSS, участвующие в приложении Thymeleaf. Подобно режиму JAVASCRIPT, режим шаблона CSS также является текстовым режимом и использует специальный синтаксис обработки из режима шаблона TEXT.

Режим шаблона RAW просто не будет обрабатывать шаблоны. Он предназначен для вставки нетронутых ресурсов (файлов, ответов на URL и тп.) в обрабатываемые шаблоны. Например, внешние, неконтролируемые ресурсы в формате HTML могут быть включены в шаблоны приложений, надежно зная, что любой код Thymeleaf, который эти ресурсы включает, не будет выполнен.

1.9 Описание фреймворка Spring

Spring Framework представляет собой просто контейнер внедрения зависимостей, с несколькими удобными слоями (например: доступ к базе данных, прокси, аспектно-ориентированное программирование, RPC, веб-инфраструктура MVC). Это все позволяет вам быстрее и удобнее создавать Java-приложения.

1.9.1 Структура Spring

Spring — это не один какой-то конкретный фреймворк. Это скорее общее название для целого ряда небольших фреймворков, каждый из которых выполняет какую-то свою работу, рисунок 1.

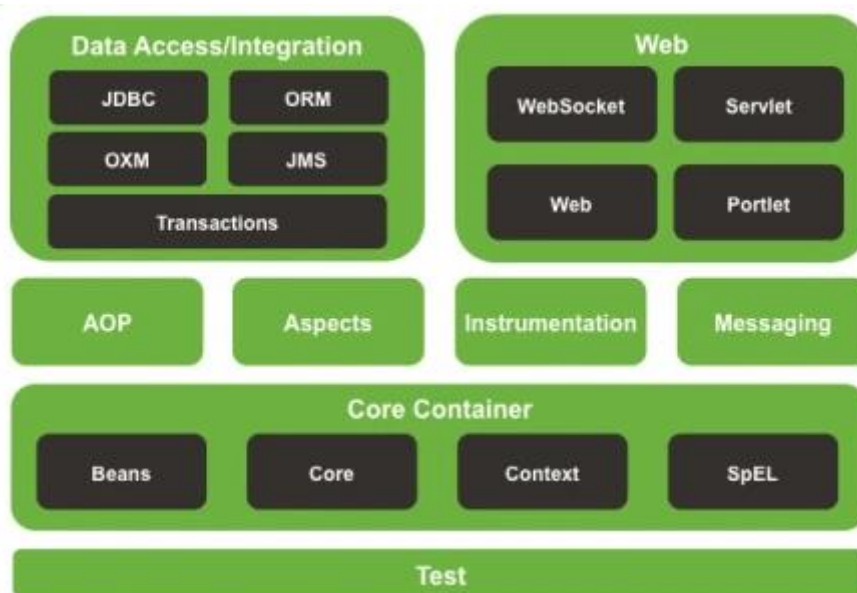


Рисунок 1 — *UML Spring*

Как видно, у спринга модульная структура. Это позволяет подключать только те модули, что нам нужны для нашего приложения и не подключать те, которыми мы заведомо не будем пользоваться. Насколько мне известно, то именно этот подход и помог спрингу обойти своего конкурента в то время (EJB) и захватить лидерство. Потому что приложения, использующие EJB тянули очень много зависимостей за собой, да и вообще получались медленные и неповоротливые.

1.10 Выводы.

Для Реализации своего приложения я выбираю Gradle для управления зависимостей, т.к он прост в изучении и управлении, пакетный менеджер Yarn за быстроту и надежность, язык Java с использованием фреймворка Spring для backend за огромное количество возможностей для разработки и с большим объемом документации, JavaScript с фреймворком Vue, за большой потенциал в будущем и самый быстро развивающийся

фреймворк на мой взгляд, с простым в освоении синтаксисом, а так же СУБД *PostgreSQL* за возможность выбрать свой интерфейс и работы с json. Аппаратная платформа Arduino nano и датчик ВМЕ 280 за компактность, но при этом со всеми возможностями, которые нужны для реализации поставленной задачи.

					ПГУ 2.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		32

2 РЕАЛИЗАЦИЯ

В данной части работы приведены результаты разработки программы сбора показателей погодных условий и реализация клиентсерверного приложения, используя REST API, Spring, Vue.

Процесс разработки этого компонента можно разделить на следующие этапы:

- подключение датчика к arduino
- разработка скетча для Arduino Nano и сбор информации с датчика BME280 и отправка в Serial Port
- разработка Java кода для работы с Arduino и получение информации из Serial port
- разработка взаимодействия с Базой данных, для записи данных и вывода уже имеющихся данных.
- разработка Rest API
- разработка страницы браузера для вывода последних данных, записанные в БД и вывод показателей в данный момент.

2.1 Подключение датчика к плате

Для подключения понадобятся сам датчик BMP280, плата Ардуино Nano, соединительные провода. Схема подключения показана на рисунке 3.

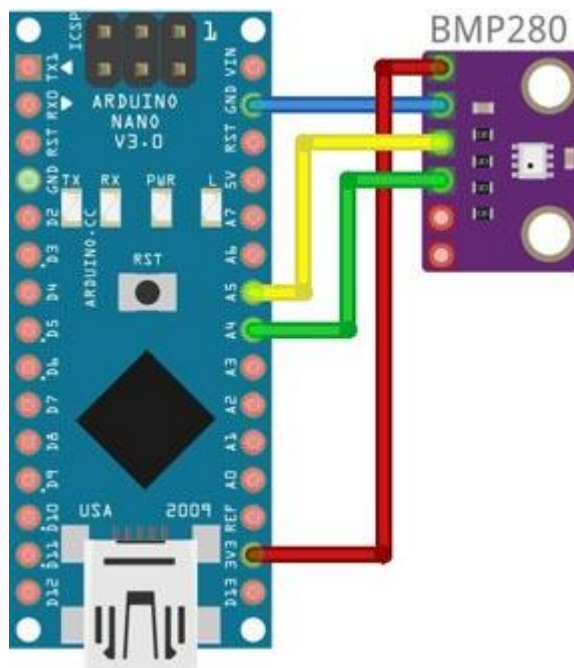


Рисунок 2 — Схема подключения датчика к плате

Для соединения с Ардуино нужно подключить выходы следующим образом: соединить землю с Ардуино и на датчике, VCC – на 3,3В, SCL / SCK – к аналоговому контакту A5, SDA / SDI – к A4.

2.2 Разработка прошивки датчика погоды

Для написания кода для Arduino используется специальная среда разработки Arduino IDE с языком программирования C, C++. Для работы с датчик BME280 используем библиотеку Adafruit_BMP280.h. Библиотека загружается в IDE из Менеджера Библиотек. Она позволяет максимально сократить время на освоение данного модуля, не урезая его функционал. Методы библиотеки дают возможность пользователю выбрать способ подключения, а также настроить периодичность и точность измерений в зависимости от режимов работы, три функции, с помощью которых можно считать показания температуры, атмосферного давления и влажность (Листинг 1). В первую очередь в коде нужно подключить все библиотеки и инициализировать работу датчика. Скетч находится в приложении А, название файла «arduino.ino»

Листинг 1. Подключение Библиотеки и настройка датчика.

```
#include <Adafruit_BMP280.h>
Adafruit_BMP280 bmp;
Serial.begin(9600);
bmp.setSampling(
  Adafruit_BMP280::MODE_NORMAL, // Режим работы
  Adafruit_BMP280::SAMPLING_X2, // Точность изм. температуры
  Adafruit_BMP280::SAMPLING_X16, // Точность изм. давления
  Adafruit_BMP280::FILTER_X16, // Уровень фильтрации
  Adafruit_BMP280::STANDBY_MS_500); // Период просыпания, мСек
Serial.print(bmp.readTemperature()); // Функция измерения температуры
Serial.print(bmp.readPressure()); // Функция измерения атм. Давления
Serial.print(bmp.readHumidity()); // Функция измерения влажности
```

2.3 Проектирование серверной части

Для этого нам понадобилась библиотека «jSerialComm», а также для удобства в application.properties я задали 2 параметра: portDescription, название порта, на котором находит Arduino и baud_rate с той же величиной, что указана в скетче у Arduino (Листинг 2). Используя возможности фреймворка Spring для метода **isConnected** из класса **StartArduino** указывается аннотация **@PostConstructor**, используя значения из application.properties, мы создаем соединение с портом на котором находит Arduino, если это возможно.

Листинг 2. Установка соединения с ComPort.

```
@PostConstruct
public boolean isConnected() throws InterruptedException {
    this.scanner = new Scanner(System.in);
    this.arduino = new Arduino(portDescription, baud_rate);
    this.connected = arduino.openConnection();
    System.out.println("Соединение с Arduino установлено: " + connected);
    Thread.sleep(2000);
    return connected;
}

public Arduino(String portDescription, int baud_rate) {
    //preferred constructor
    this.portDescription = portDescription;
    comPort = SerialPort.getCommPort(this.portDescription);
    this.baud_rate = baud_rate;
    comPort.setBaudRate(this.baud_rate);
}

public boolean openConnection(){
    if(comPort.openPort()){
        try {Thread.sleep(100);} catch(Exception e){}
        return true;
    }
    else {
        AlertBox alert = new AlertBox(new Dimension(400,100),"Error
Connecting", "Try Another port");
        alert.display();
        return false;
    }
}
```

2.4 Проектирование базы данных разрабатываемой системы

При разработке системы была использована СУБД *PostgreSQL*. Далее представлено описание структуры, разработанной БД. Таблица 1 содержит описание структуры таблицы «weather».

Таблица 1 — Структура таблицы «weather».

Название поля	Описание поля	Тип данных
id	Идентификатор	Long
temperature	Температура	int
pressure	Давление	Int
humidity	Влажность	Int
creationDate	Время создания	Date

Настройка подключения к БД. Мы используем *spring jpa*. Для соединения с бд нужно использовать файл *application.properties*. (Листинг 3) *url*-адрес нашей БД, *username* – пользователь БД. *Password* – пароль пользователя, *generate-ddl* – используются для автоматического создания таблицы в бд, используя нашу сущность.

Листинг 3. Функция отправки сообщения:

```
spring.datasource.url=jdbc:postgresql://localhost:5000/akkuratnovWeather
spring.datasource.username=postgres
spring.datasource.password=123
spring.jpa.generate-ddl=true

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

Класс **Weather.java** имеет 6 полей: *id*, *temperature*, *humidity*, *pressure*, *creationDate*. Для связи сущности с БД используется класс *UserDatalisRepo*, с расширенной возможностью поиска по дате создания.

Листинг 4. Интерфейс взаимодействия с БД с сущностью Weather:

```
public interface WeatherRepo extends JpaRepository<Weather, Long> {

    Weather findByCreationDate(LocalDateTime localDateTime);

}
```

Если соединение установлено с *Arduino* то, используя еще одну возможность *Spring*'а для выполнения задачи по расписанию *@Scheduled(cron*

= "0 0 0,6,12,18 * * *") у метода dataCollection для получения данных с датчика в 00:00, 06:00, 12:00, 18:00, если приложение запущено и записывать из в Базу данных (Листинг 5).

Листинг 5. Запуск задачи по распианию для.

```
@Scheduled(cron = "0 0 0,6,12,18 * * *")
public void dataCollection(){
    String message = arduino.serialRead();
    String[] indicators = message.split(";");
    Weather weather = new Weather();
    weather.setTemperature(Integer.parseInt(indicators[0]));
    weather.setPressure(Integer.parseInt(indicators[1]));
    weather.setHumidity(Integer.parseInt(indicators[2]));
    messageRepo.saveAndFlush(weather);
}
```

2.5 Разработка Rest API

При разработке Rest API был создан только один и основной GetMapping”(Листинг 6). Для вывода информации о состояниях датчик за последние 24 часа и в данный момент, мы создадим самое просто одностраничное приложение. В ответе на запрос мы через thymeleaf будем передавать 2 параметра weathers и now. В Weathers массив из последних 5 записей в БД, now - текущие показатели датчика.

Листинг 6. Rest API GetMapping””

```
@GetMapping
public String main(Model model) {
    HashMap<Object, Object> data = new HashMap<>();
    List<Weather> dateBase = messageRepo.findAll();
    List<Weather> sendDB = new ArrayList<>();
    for( int i = dateBase.size() - 1; i>dateBase.size()-6;
i--){
        sendDB.add(dateBase.get(i));
    }
    StartArduino arduino = (StartArduino)
context.getBean("startArduino");
    String[] indicators = arduino.readMessage().split(";");
    Weather weather = new Weather();
    weather.setTemperature(Integer.parseInt(indicators[0]));
    weather.setPressure(Integer.parseInt(indicators[1]));
    weather.setHumidity(Integer.parseInt(indicators[2]));//
    data.put("now", weather);
    data.put("weathers", sendDB);
    model.addAttribute("frontendData", data);
    return "index";
}
```

2.6 Разработка Web-страницы

Для реализации страницы браузеры мы используем html, js, Vue фреймворк с Material Design Vuetify. В файле index.html, который возвращает GetMapping"/", находится только 1 div с id="app". За рендер отвечает скрипт main.js. Vue рекомендует делить все содержимое страницы на компоненты, затем использовать их в нужных местах. Файл App.vue(Листинг 6) самый главный компонент, он и рендерится в index.html в <div id= "app">. Так же было создано еще 3 компонента. Today.vue – компонент отображающий текущие показатели датчика. WeatherList.vue и WeatherRow.vue для отображения списка последних записей в Базе Данных.

Листинг 7. App.vue template

```
<v-app>
  <v-app-bar app>
    <v-toolbar-title>Порода</v-toolbar-title>
    <v-spacer></v-spacer>
    Аккуратнов Иван
  </v-app-bar>
  <v-content class="d-inline" >
    <v-layout>
      <v-responsive width="43%" max-width="43%" class="deep-purple
darken-4">
        <today></today>
      </v-responsive>
      <v-responsive>
        <weather-list :weathers="weathers" />
      </v-responsive>
    </v-layout>
  </v-content>
  <v-footer>&copy; 2020</v-footer>
</v-app>
```

2.7 Взаимодействие классов

– В системе по отправке и приему сообщений для взаимодействия ранее описанных СУБД и REST API архитектура разделена на основные функциональные блоки:

- классы-контроллеры – выполняют запросы *REST API*;
- классы-сущности – отвечают за описание сущностей БД в программе.

- классы-репозитории – выполняют взаимодействие СУБД с сущностями. Данные передаются в эти классы из классов-сервисов;
- классы компоненты - создаются один раз при запуске программы, для выполнения определенных методов

UML диаграмма классов серверной части представлена на рисунке 3.

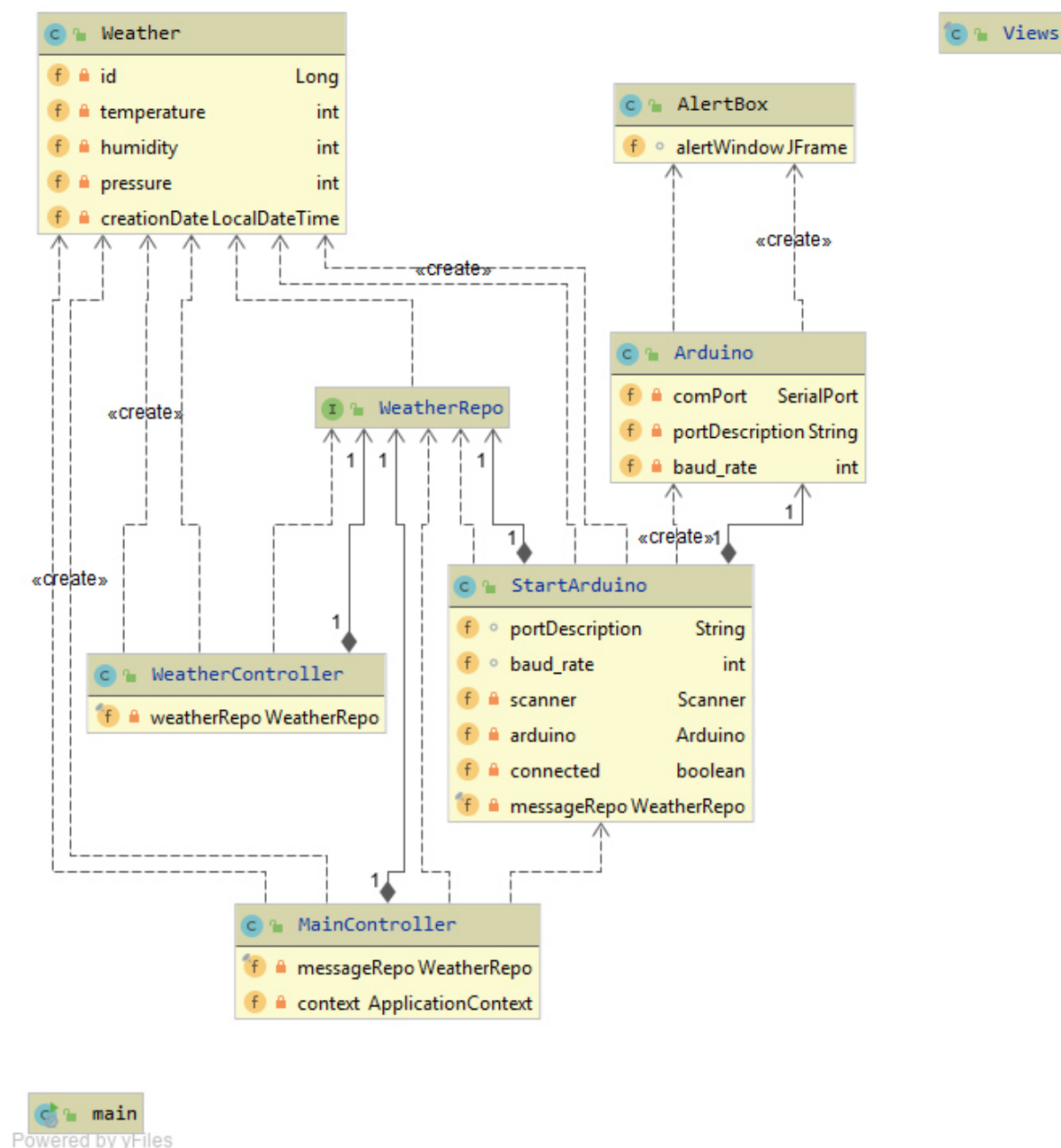


Рисунок 3 — *UML* диаграмма классов сервера

UML диаграмма, показывающая структуру frontend, представлена на рисунке 4.

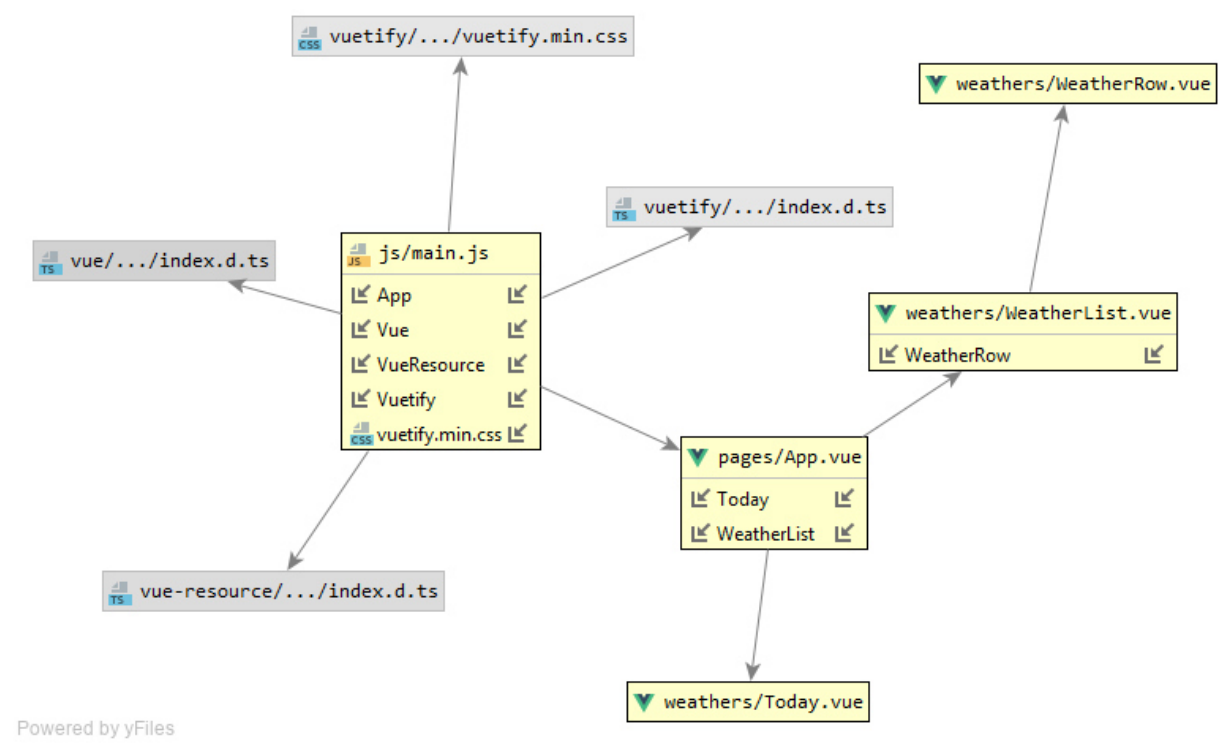


Рисунок 4 — UML диаграмма frontend

3 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

3.1 Результат работы приложения со стороны пользователя системы

Для пользователя использовать приложение максимально просто. Перейти в браузер на страницу приложения. Там ему покажет текущее состояние датчик и последние показатели, сделанные по расписанию и занесенные в базу данных, рисунок 5.

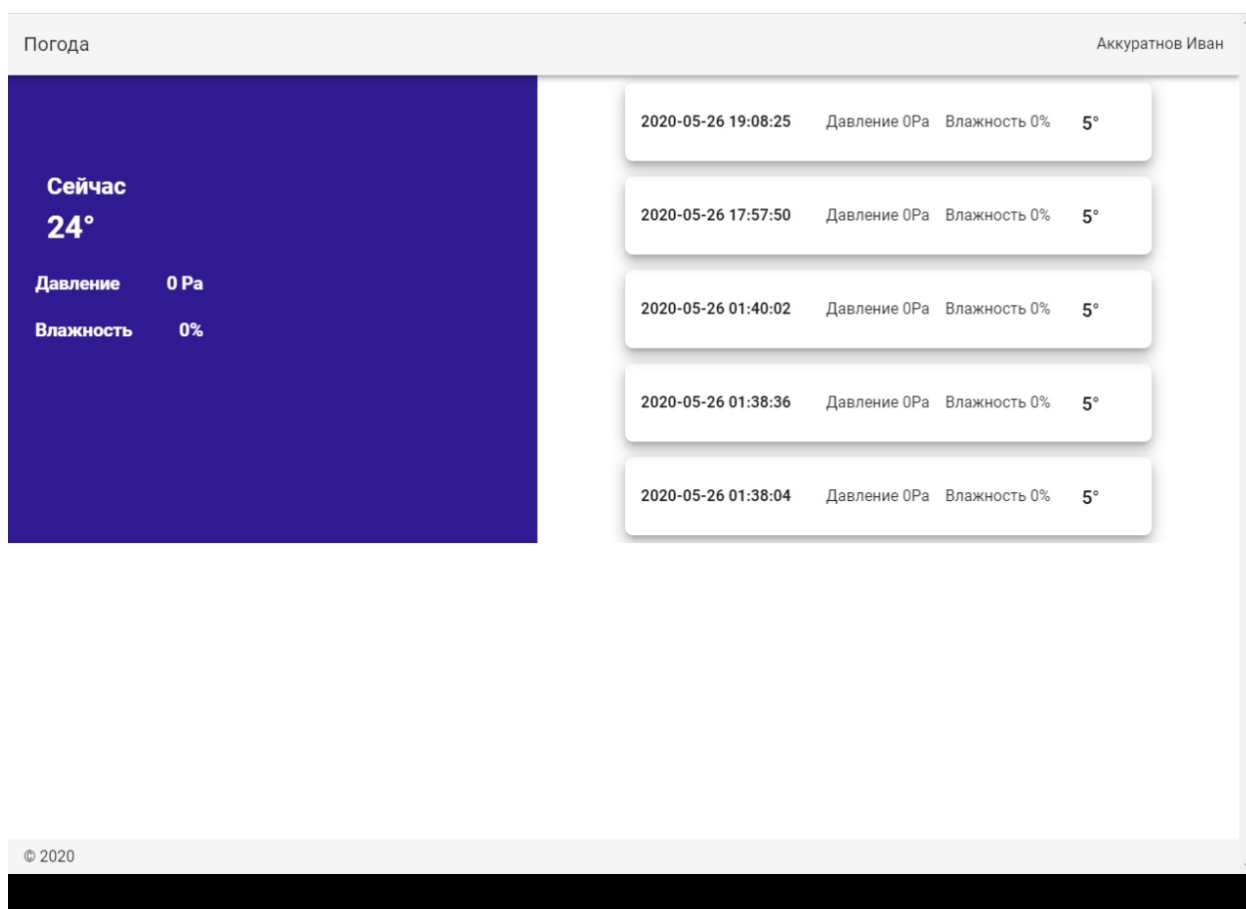


Рисунок 5 — Web-страница

Запуск программы состоит из 2 частей: Сервера Java (рисунок 6).

Изм.	Лист	№ докум.	Подпись	Дата

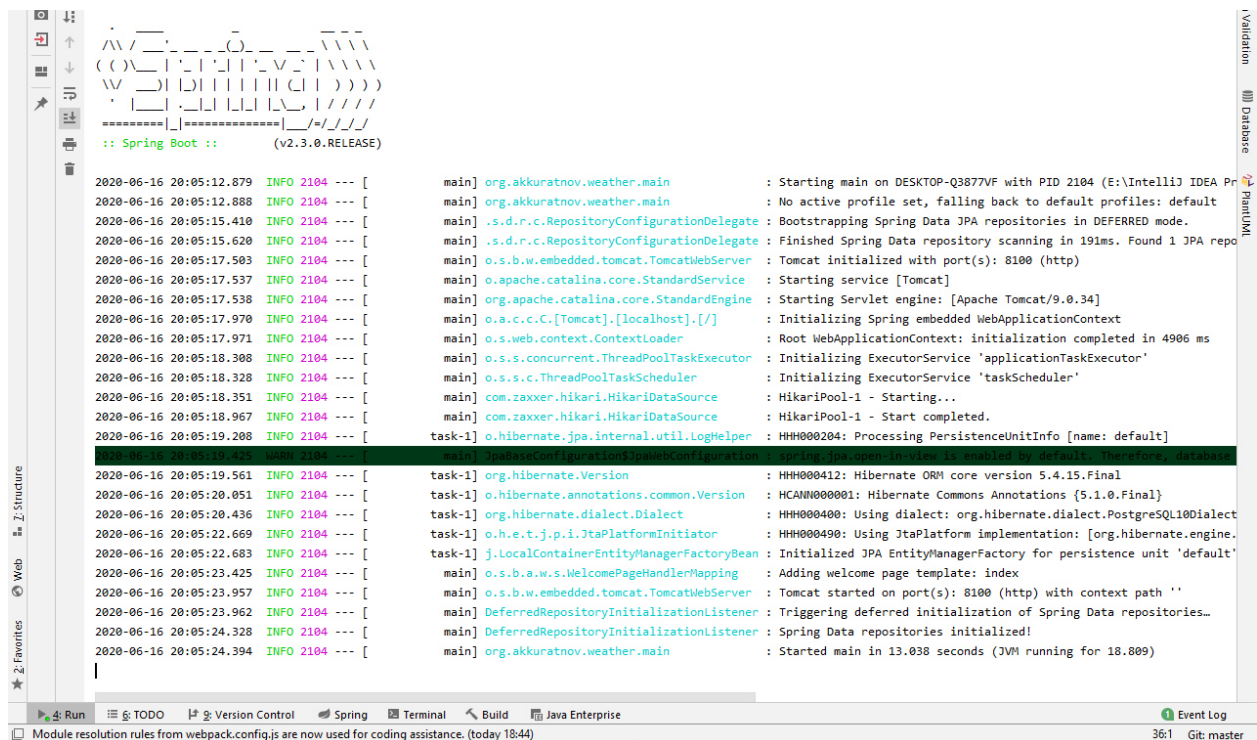


Рисунок 6 — Лого запуска сервера

Запуск webPack командой “npm start” (рисунок 7).

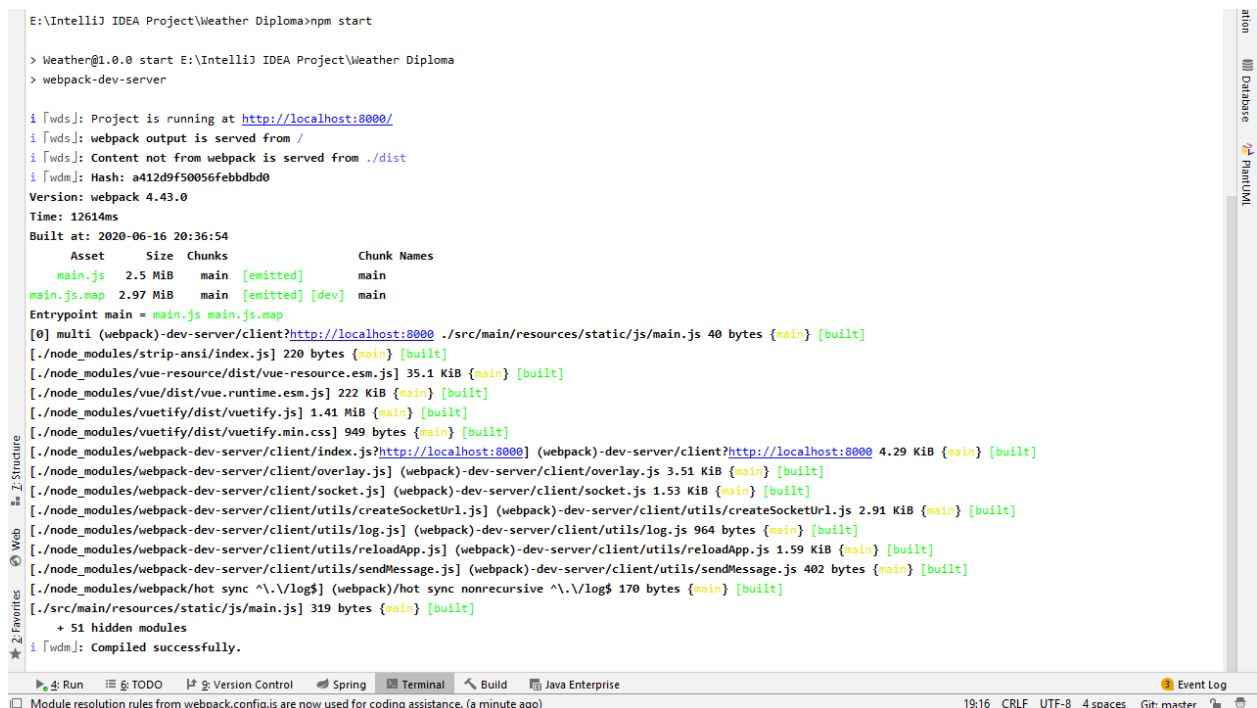


Рисунок 7 — Лого запуска WebPack

Таблица weather создастся автоматически при первой попытке записи сущности в Базу данных “akkuratnovWeather”, рисунок 8.

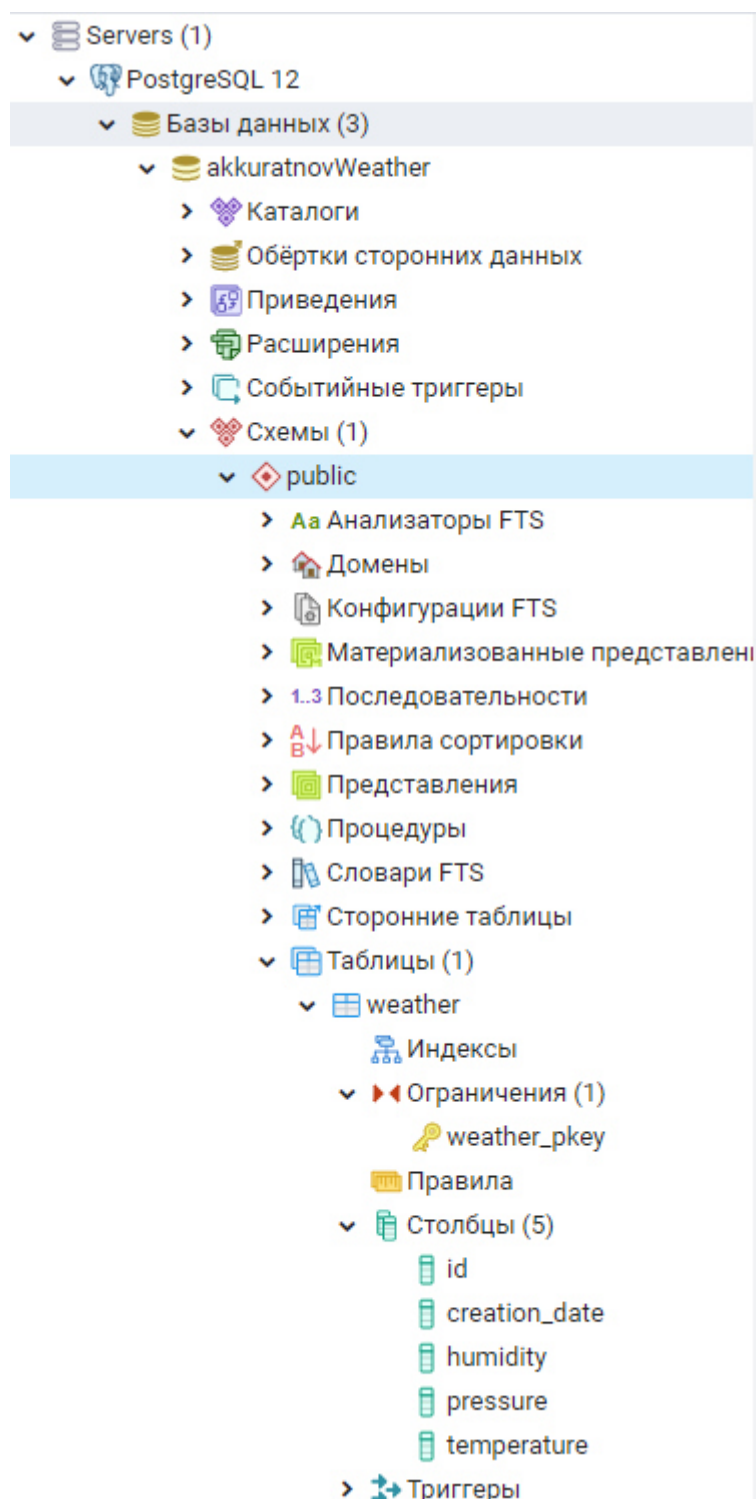


Рисунок 8 — Содержимое базы данных akkuratnovWeather

Формат получения данных от сервера через thymeleaf в Console браузера, рисунок 9.



Рисунок 9 — Содержимое ответа от сервера с данными

3.2 Выводы

Результатом разработки клиент-серверного приложения Погода, для клиента это страница в браузере с необходимой ему информацией для мониторинга его Датчика погоды, установленного в определенном месте, система сбора информации с датчика, и сервер отвечающий за весь необходимый функционал приложения.

					ПГУ 2.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		45

4 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

С развитием научно-технического прогресса немаловажную роль играет возможность безопасного исполнения людьми своих трудовых обязанностей. В связи с этим была создана и развивается наука о безопасности труда и жизнедеятельности человека.

Безопасность жизнедеятельности (БЖД) - это комплекс мероприятий, направленных на обеспечение безопасности человека в среде обитания, сохранение его здоровья, разработку методов и средств защиты путем снижения влияния вредных и опасных факторов до допустимых значений, выработку мер по ограничению ущерба в ликвидации последствий чрезвычайных ситуаций мирного и военного времени.

Цель и содержание БЖД:

- обнаружение и изучение факторов окружающей среды, отрицательно влияющих на здоровье человека;
- ослабление действия этих факторов до безопасных пределов или исключение их если это возможно;
- ликвидация последствий катастроф и стихийных бедствий.

Круг практических задач БЖД прежде всего обусловлен выбором принципов защиты, разработкой и рациональным использованием средств защиты человека и природной среды от воздействия техногенных источников и стихийных явлений, а также средств, обеспечивающих комфортное состояние среды жизнедеятельности.

Охрана здоровья трудящихся, обеспечение безопасности условий труда, ликвидация профессиональных заболеваний и производственного травматизма составляет одну из главных забот человеческого общества. Обращается внимание на необходимость широкого применения прогрессивных форм научной организации труда, сведения к минимуму

ручного, малоквалифицированного труда, создания обстановки, исключающей профессиональные заболевания и производственный травматизм [21].

На рабочем месте должны быть предусмотрены меры защиты от возможного воздействия опасных и вредных факторов производства. Уровни этих факторов не должны превышать предельных значений, оговоренных правовыми, техническими и санитарно-техническими нормами. Эти нормативные документы обязывают к созданию на рабочем месте условий труда, при которых влияние опасных и вредных факторов на работающих либо устранено совсем, либо находится в допустимых пределах.

Данный раздел дипломного проекта посвящен рассмотрению следующих вопросов:

- определение оптимальных условий труда инженера - программиста;
- расчет освещенности;
- расчет уровня шума.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была изучена методология проектирования приложения с использованием Spring, СУБД PostgreSQL, VUE. Использование таких библиотек как jSerialComm, spring jpa, thymeleaf, vuetify. При реализации поставленной задачи было разработано и отлажено клиент-серверное приложение «Погода» с использованием основных языков Java и JavaScript, C, C++.

					ПГУ 2.090301.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Николай Прохоренок, Владимир Дронов. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера. — СПб.: «БХВ-Петербург», 2015. — С. 20-348.
2. Леон Шкляр, Рич Розен. Архитектура веб-приложений. — М.: «Эксмо», 2010. — С.35-150.
3. Эрик Фримен, Элизабет Робсон. Изучаем программирование на JavaScript. — СПб.: «Питер», 2018. — С.125-340.
4. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. — «Springer Science & Business Media», 2011. — С. 52-340.
5. Banks A., Porcello E. Learning React: functional web development with React and Redux. — " O'Reilly Media, Inc.", 2017.
6. Трепачёв Д.П. Учебник по фреймворку VUE //Аллея науки. – 2020. – Т. 1. – №. 7. – С. 928-930.
7. Дж. Уорсли, Дж. Дрейк. PostgreSQL . Для профессионалов — СПб.: «Питер», 2003. — С. 23-320.
8. Хэррон Д. Node. js Разработка серверных веб-приложений на JavaScript. – Litres, 2017.
9. Ефремова О. С. Требования охраны труда при работе на персональных электронно-вычислительных машинах. – М. : Альфа-Пресс, 2008.

ПРИЛОЖЕНИЯ

Приложение А Листинги программы на JavaScript

Файл «arduino.ino»

```
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP280.h>

#define BMP_SCK  (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS   (10)

Adafruit_BMP280 bmp; // I2C
//Adafruit_BMP280 bmp(BMP_CS); // hardware SPI
//Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO,  BMP_SCK);

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

  if (!bmp.begin()) {
    Serial.println(F("Could not find a valid BMP280 sensor,
check wiring!"));
    while (1);
  }

  /* Default settings from datasheet. */
  bmp.setSampling(Adafruit_BMP280::MODE_NORMAL, /* Operating
Mode. */
                 Adafruit_BMP280::SAMPLING_X2, /* Temp.
oversampling */
                 Adafruit_BMP280::SAMPLING_X16, /* Pressure
oversampling */
                 Adafruit_BMP280::FILTER_X16, /*
Filtering. */
                 Adafruit_BMP280::STANDBY_MS_500); /* Standby
time. */
}

void loop() {
  //Serial.print(F("Temperature = "));
  Serial.print(F("!"));
  Serial.print(bmp.readTemperature());
  Serial.println(";");

  //Serial.print(F("Pressure = "));
  Serial.print(bmp.readPressure());
  Serial.println(";");
}
```

```

//Serial.print(F("Approx altitude = "));
Serial.print(bmp.readhumidity());
Serial.println("$");

Serial.println();
delay(2000);
}

```

Файл «build.gradle»

```

plugins {
    id 'java'
    id "org.springframework.boot" version "2.2.6.RELEASE"
}

group 'weather'
version '1.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    testCompile group: 'junit', name: 'junit', version: '4.12'
    //Для порта Ардуина
    compile group: 'com.fazecast', name: 'jSerialComm', version:
'2.6.2'

    compile group: 'org.springframework.boot', name: 'spring-
boot-starter-parent', version: '2.2.7.RELEASE', ext: 'pom'
    compile group: 'org.springframework.boot', name: 'spring-
boot-starter-web', version: '2.2.7.RELEASE'

    //Для Базы данных
    compile group: 'org.springframework.boot', name: 'spring-
boot-starter-data-jpa', version: '2.3.0.RELEASE'
    compile group: 'org.postgresql', name: 'postgresql',
version: '42.2.12'
    compile group: 'javax.xml.bind', name: 'jaxb-api', version:
'2.3.1'
    compile group: 'org.glassfish.jaxb', name: 'jaxb-core',
version: '2.3.0.1'
    compile group: 'javax.xml', name: 'jaxb-impl', version:
'2.1'
    compile group: 'org.projectlombok', name: 'lombok', version:
'1.18.10'

```

Изм.	Лист	№ докум.	Подпись	Дата

```
    compile group: 'org.springframework.boot', name: 'spring-
boot-starter-thymeleaf', version: '2.3.0.RELEASE'
```

```
}
```

Файл «package.json»

```
{
  "name": "Weather",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "start": "webpack-dev-server"
  },
  "dependencies": {
    "vue": "^2.6.11",
    "vue-resource": "^1.5.1",
    "vuetify": "^2.2.30"
  },
  "devDependencies": {
    "@babel/core": "^7.9.6",
    "@babel/preset-env": "^7.9.6",
    "babel-loader": "^8.1.0",
    "components": "^0.1.0",
    "css-loader": "^3.5.3",
    "vue-loader": "^15.9.2",
    "vue-style-loader": "^4.1.2",
    "vue-template-compiler": "^2.6.11",
    "webpack": "^4.43.0",
    "webpack-cli": "^3.3.11",
    "webpack-dev-server": "^3.11.0"
  }
}
```

Файл «webpack.config.js»

```
const path = require('path');
const VueLoaderPlugin = require('vue-loader/lib/plugin');

module.exports = {
  mode: 'development',
  devtool: 'source-map',
  entry: path.join(__dirname, 'src', 'main', 'resources',
'static', 'js', 'main.js'),
  devServer: {
    contentBase: './dist',
    compress: true,
    port: 8000,
    allowedHosts: [
      'localhost:8100'
    ]
  }
}
```

```

    ],
    stats: 'errors-only',
    clientLogLevel: 'error'
  },
  module: {
    rules: [
      {
        test: /\.js$/,
        exclude: /(node_modules|bower_components)/,
        use: {
          loader: 'babel-loader',
          options: {
            presets: ['@babel/preset-env']
          }
        }
      },
      {
        test: /\.vue$/,
        loader: 'vue-loader'
      },
      {
        test: /\.css$/,
        use: [
          'vue-style-loader',
          'css-loader'
        ]
      }
    ]
  },
  plugins: [
    new VueLoaderPlugin()
  ],
  resolve: {
    modules: [
      path.join(__dirname, 'src', 'main', 'resources',
        'static', 'js'),
      path.join(__dirname, 'node_modules'),
    ],
  },
}

```

Файл «application.properties»

```

spring.portDescription = "COM3"
spring.baud_rate = 9600
server.port = 8100

```

```

spring.datasource.url=jdbc:postgresql://localhost:5000/akkuratno
vWeather
spring.datasource.username=postgres
spring.datasource.password=123

```

Изм.	Лист	№ докум.	Подпись	Дата

```
spring.jpa.generate-ddl=true
```

```
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation  
=true
```

Файл «Arduino.java»

```
package org.akkuratnov.weather.arduino;
```

```
import java.awt.Dimension;  
import java.io.PrintWriter;  
import java.util.Scanner;
```

```
import com.fazecast.jSerialComm.*;
```

```
public class Arduino {  
    private SerialPort comPort;  
    private String portDescription;  
    private int baud_rate;  
  
    public Arduino() {  
        //empty constructor if port undecided  
    }  
    public Arduino(String portDescription) {  
        //make sure to set baud rate after  
        this.portDescription = portDescription;  
        comPort = SerialPort.getCommPort(this.portDescription);  
    }  
  
    public Arduino(String portDescription, int baud_rate) {  
        //preferred constructor  
        this.portDescription = portDescription;  
        comPort = SerialPort.getCommPort(this.portDescription);  
        this.baud_rate = baud_rate;  
        comPort.setBaudRate(this.baud_rate);  
    }  
  
    public boolean openConnection(){  
        if(comPort.openPort()){  
            try {Thread.sleep(100);} catch(Exception e){}  
            return true;  
        }  
        else {  
            AlertBox alert = new AlertBox(new  
Dimension(400,100),"Error Connecting", "Try Another port");  
            alert.display();  
            return false;  
        }  
    }  
}
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 2.090301.01.001 ПЗ

Лист

54

```

    }

    public void closeConnection() {
        comPort.closePort();
    }

    public void setPortDescription(String portDescription){
        this.portDescription = portDescription;
        comPort = SerialPort.getCommPort(this.portDescription);
    }

    public void setBaudRate(int baud_rate){
        this.baud_rate = baud_rate;
        comPort.setBaudRate(this.baud_rate);
    }

    public String getPortDescription(){
        return portDescription;
    }

    public SerialPort getSerialPort(){
        return comPort;
    }

    public String serialRead(){
        //will be an infinite loop if incoming data is not bound

        comPort.setComPortTimeouts(SerialPort.TIMEOUT_READ_SEMI_BLOCKING
        , 0, 0);
        String out="";
        String key = "";
        Scanner in = new Scanner(comPort.getInputStream());
        try
        {
            while (in.hasNext())
                if (in.next().equals("!"))
                    break;

            while(in.hasNext()) {
                key = in.next();
                if(!key.equals("$"))
                    out += (key);
                else break;
            }
            in.close();
        } catch (Exception e) { e.printStackTrace(); }
        return out;
    }

    public String serialRead(int limit){

```

```

        //in case of unlimited incoming data, set a limit for
        number of readings

comPort.setComPortTimeouts(SerialPort.TIMEOUT_READ_SEMI_BLOCKING
, 0, 0);
    String out="";
    int count=0;
    Scanner in = new Scanner(comPort.getInputStream());
    try
    {
        while(in.hasNext() && count<=limit) {
            out += (in.next()+"\n");
            count++;
        }
        in.close();
    } catch (Exception e) { e.printStackTrace(); }
    return out;
}

public void serialWrite(String s){
    //writes the entire string at once.
    comPort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER,
0, 0);
    try{Thread.sleep(5);} catch(Exception e){}
    PrintWriter pout = new
PrintWriter(comPort.getOutputStream());
    pout.print(s);
    pout.flush();

}

public void serialWrite(String s,int noOfChars, int delay){
    //writes the entire string at once.
    comPort.setComPortTimeouts(SerialPort.TIMEOUT_SCANNER,
0, 0);
    try{Thread.sleep(5);} catch(Exception e){}
    PrintWriter pout = new
PrintWriter(comPort.getOutputStream());
    for(int i=0;i<s.length();i+=noOfChars){
        pout.write(s.substring(i,i+noOfChars));
        pout.flush();
        System.out.println(s.substring(i,i+noOfChars));
        try{Thread.sleep(delay);} catch(Exception e){}
    }
    pout.write(noOfChars);
    pout.flush();

}

public void serialWrite(char c){
    //writes the entire string at once.

```

```

        comPort.setComPortTimeouts (SerialPort.TIMEOUT_SCANNER,
0, 0);
        try{Thread.sleep(5);} catch(Exception e){}
        PrintWriter pout = new
PrintWriter(comPort.getOutputStream());pout.write(c);
        pout.flush();
    }

    public void serialWrite(char c, int delay){
        //writes the entire string at once.
        comPort.setComPortTimeouts (SerialPort.TIMEOUT_SCANNER,
0, 0);
        try{Thread.sleep(5);} catch(Exception e){}
        PrintWriter pout = new
PrintWriter(comPort.getOutputStream());pout.write(c);
        pout.flush();
        try{Thread.sleep(delay);}catch(Exception e){}
    }
}

```

Файл «AlertBox.java»

```

package org.akkuratnov.weather.arduino;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AlertBox {
    JFrame alertWindow;
    public AlertBox(Dimension obj, String title, String message)
    {
        alertWindow = new JFrame();
        alertWindow.setTitle(title);
        alertWindow.setSize(obj);
        alertWindow.setPreferredSize(obj);
        alertWindow.setLayout(new BorderLayout());
        alertWindow.setLocationRelativeTo(null);
        JLabel lblMessage = new JLabel(message,
SwingConstants.CENTER);
        JButton btnOk = new JButton("Ok");
        btnOk.addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                alertWindow.setVisible(false);
                alertWindow.dispose();
            }

        });
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

```

        alertWindow.add(btnOk, BorderLayout.SOUTH);
        alertWindow.add(lblMessage, BorderLayout.CENTER);
    }

    public void display(){
        alertWindow.setVisible(true);
    }
}

```

Файл «StartArduino.java»

```

package org.akkuratnov.weather.arduino;

import java.util.Scanner;

import org.springframework.beans.factory.annotation.*;
import org.springframework.stereotype.Component;

import javax.annotation.*;

@Component
public class StartArduino {

    @Value("${spring.portDescription}")
    String portDescription;
    @Value("${spring.baud_rate}")
    int baud_rate;
    Scanner scanner;
    Arduino arduino;
    boolean connected;

    @PostConstruct
    public boolean isConnected() throws InterruptedException {
        this.scanner = new Scanner(System.in);
        this.arduino = new Arduino(portDescription, baud_rate);
        this.connected = arduino.openConnection();
        System.out.println("Соединение с Arduino установлено: "
+ connected);
        Thread.sleep(2000);
        return connected;
    }

    public String readMessage(){
        return arduino.serialRead();
    }

    @PreDestroy
    public void closeArduino(){
        arduino.closeConnection();
    }
}

```

Изм.	Лист	№ докум.	Подпись	Дата

```

@Scheduled(cron = "0 0 6,19 * * *")
public void dataCollection(){
    String message = arduino.serialRead();
    String[] indicators = message.split(";");
    Weather weather = new Weather();
    weather.setTemperature(Integer.parseInt(indicators[0]));
    weather.setPressure(Integer.parseInt(indicators[1]));
    weather.setHumidity(Integer.parseInt(indicators[2]));
    messageRepo.saveAndFlush(weather);
}

}

```

Файл «main.java»

```

package org.akkuratnov.weather;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class main {
    public static void main(String[] args) throws
InterruptedException {
        SpringApplication.run(main.class, args);
    }
}

```

Файл «MainController.java»

```

package org.akkuratnov.weather.controllers;

import org.akkuratnov.weather.arduino.StartArduino;
import org.akkuratnov.weather.domain.Weather;
import org.akkuratnov.weather.repo.WeatherRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.ApplicationContext;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

@Controller
@RequestMapping({"", "/"})

```

```

public class MainController {
    private final WeatherRepo messageRepo;

    @Autowired
    private ApplicationContext context;

    @Autowired
    public MainController(WeatherRepo messageRepo) {
        this.messageRepo = messageRepo;
    }

    @GetMapping
    public String main(Model model) {
        HashMap<Object, Object> data = new HashMap<>();
        List<Weather> dateBase = messageRepo.findAll();
        List<Weather> sendDB = new ArrayList<>();
        for( int i = dateBase.size() - 1; i>dateBase.size()-6;
i--){
            sendDB.add(dateBase.get(i));
        }
        StartArduino arduino = (StartArduino)
context.getBean("startArduino");
        String[] indicators = arduino.readMessage().split(";");
        Weather weather = new Weather();
        weather.setTemperature(Integer.parseInt(indicators[0]));
        weather.setPressure(Integer.parseInt(indicators[1]));
        weather.setHumidity(Integer.parseInt(indicators[2]));//
        data.put("now", weather);
        data.put("weathers", sendDB);
        model.addAttribute("frontendData", data);
        return "index";
    }

}

```

Файл «Weather.java»

```

package org.akkuratnov.weather.domain;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonView;
import lombok.EqualsAndHashCode;
import lombok.ToString;

import javax.persistence.*;
import java.time.LocalDateTime;

@Entity
@Table

```

```

@Override(of = {"id", "temperature", "humidity", "pressure"})
@EqualsAndHashCode(of = {"id"})
public class Weather {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @JsonView(Views.IdTempTime.class)
    private Long id;
    @JsonView(Views.IdTempTime.class)
    private int temperature;
    @JsonView(Views.FullMessage.class)
    private int humidity;
    @JsonView(Views.FullMessage.class)
    private int pressure;

    @Column(uptdatable = false)
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern =
"yyyy-MM-dd HH:mm:ss")
    @JsonView(Views.IdTempTime.class)
    private LocalDateTime creationDate;

    public Long getId(){
        return id;
    }

    public int getTemperature(){
        return temperature;
    }

    public void setTemperature(int temperature){
        this.temperature = temperature;
    }

    public int getHumidity(){
        return humidity;
    }

    public void setHumidity(int humidity){
        this.humidity = humidity;
    }

    public int getPressure(){
        return pressure;
    }

    public void setPressure(int pressure){
        this.pressure = pressure;
    }

    public LocalDateTime getCreationDate(){
        return creationDate;
    }
}

```

```

    }

    public void setCreationDate(LocalDateTime creationDate){
        this.creationDate = creationDate;
    }
}

```

Файл «arduino.ino»

```

package org.akkuratnov.weather.repo;

import org.akkuratnov.weather.domain.Weather;
import org.springframework.data.jpa.repository.JpaRepository;

import java.time.LocalDateTime;

public interface WeatherRepo extends JpaRepository<Weather,
Long> {
    Weather findByCreationDate(LocalDateTime localDateTime);
}

```

Файл «index.html»

```

<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <link
href="https://fonts.googleapis.com/css?family=Roboto:100,300,400
,500,700,900" rel="stylesheet">
    <meta name="viewport" content="width=device-width, initial-
scale=1, maximum-scale=1, user-scalable=no, minimal-ui">
    <link
href="https://cdn.jsdelivr.net/npm/@mdi/font@5.x/css/materialdes
ignicons.min.css" rel="stylesheet">
    <link
href="https://cdn.jsdelivr.net/npm/vuetify@2.x/dist/vuetify.min.
css" rel="stylesheet">
    <script th:inline="javascript">
        var frontendData = [{${frontendData}}];
    </script>
</head>
<body>
<div id="app"></div>
<script th:src="'http://localhost:8000/main.js'"></script>
</body>
</html>

```

Файл «main.js»

```
import Vue from 'vue'
import VueResource from 'vue-resource'
import App from '../js/pages/App.vue'
import Vuetify from 'vuetify'
import 'vuetify/dist/vuetify.min.css'
```

```
Vue.use(Vuetify)
Vue.use(VueResource)
```

```
new Vue({
  el: '#app',
  vuetify: new Vuetify(),
  render: a => a(App)
})
```

Файл «App.vue»

```
<template>
  <v-app>
    <v-app-bar app>
      <v-toolbar-title>Порода</v-toolbar-title>
      <v-spacer></v-spacer>
      Аккуратнов Иван
    </v-app-bar>
    <v-content class="d-inline" >
      <v-layout>
        <v-responsive width="43%" max-width="43%"
class="deep-purple darken-4">
          <today></today>
        </v-responsive>
        <v-responsive>
          <weather-list :weathers="weathers" />
        </v-responsive>
      </v-layout>
    </v-content>
    <v-footer>&copy; 2020</v-footer>
  </v-app>
</template>
```

```
<script>
  import WeatherList from
  '../components/weathers/WeatherList.vue'
  import Today from '../components/weathers/Today.vue'
  export default {
    components: {
      WeatherList,
      Today
    },
    data() {
      return{
```

```

        weathers: frontendData.weathers
      }
    }
  }
</script>

<style>

</style>

```

Файл «Today.vue»

```

<template>
  <div>
    <div class="white--text c--h1 mt-12 ml-10 pt-12 ">
      <h2>Сейчас</h2>
      <h1>24°</h1>
      <v-layout class="mt-5" row>
        <h3 class="mu-3">Давление</h3>
        <h3 class="ml-12">0 Pa</h3>
      </v-layout>
      <v-layout class="mt-5" row>
        <h3 class="mu-3">Влажность</h3>
        <h3 class="ml-12">0%</h3>
      </v-layout>
      <!--      {{ now.temperature }}°-->
      <!--      Давление {{ now.pressure }}Pa-->
      <!--      Влажность {{ now.humidity }}%-->
    </div>
  </div>
</template>

<script>
  export default {
    name: "today",
    data() {
      return{
        now: frontendData.now
      }
    }
  }
</script>

<style scoped>

</style>

```

Файл «WeatherList.vue»

```

<template>

```

Изм.	Лист	№ докум.	Подпись	Дата

```

    <v-layout align-center justify-center column>
      <weather-row v-for="weather in weathers" :key="weather.id"
:weather="weather"/>
    </v-layout>
  </template>

<script>
  import WeatherRow from '../weathers/WeatherRow.vue'

  export default {
    props: ['weathers'],
    components: {
      WeatherRow
    },
    data() {
      return {
        weather: null
      }
    }
    // created: function () {
    //   weatherApi.get().then(result =>
    //     result.json().then(data =>
    //       data.forEach(weather =>
this.weathers.push(weather))
    //     )
    //   )
    // }
  }
</script>

<style>

</style>

```

Файл «WeatherRow.vue»

```

<template>
  <v-responsive class=" my-2 rounded-lg elevation-9"
width="75%" height="80">
    <v-layout row class="my-7">
      <b class="mx-7">{{ weather.creationDate }}</b>
      <div class="mx-2"> Давление {{ weather.pressure
}}Pa</div>
      <div class="mx-2"> Влажность {{ weather.humidity }}%
</div>
      <h3 class="ml-6"> {{ weather.temperature}}° </h3>
    </v-layout>
  </v-responsive>
</template>

```

Изм.	Лист	№ докум.	Подпись	Дата

```
<script>
  export default {
    props: ['weather']
  }
```

```
</script>
```

```
<style>
```

```
</style>
```

					ПГУ 2.090301.01.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В Презентация

Клиент-серверное приложение Погода

Автор работы: студент группы 16BB2 Аккуратнов И.М.

Научный руководитель: к.т.н, доцент Дубравин А.В.

Цель и объект работы

- Цель работы: разработать приложение позволяющее отслеживать состояние погодных условий в определенных местах
- Объект работы: клиент-серверное приложение выполняющее следующие функции
 - Сбор данных с датчика погодных условий
 - Хранение данных датчика полученные в определенное время
 - Показ данных за последние сутки
 - Показ данных в настоящий момент

Стек технологий



4

Arduino Nano

Микроконтроллер	ATmega328
Рабочее напряжение (логическая уровень)	5 В
Входное напряжение (рекомендуемое)	7-12 В
Входное напряжение (предельное)	6-20 В
Цифровые Входы/Выходы	14 (6 из которых могут использоваться как выходы ШИМ)
Аналоговые входы	8
Постоянный ток через вход/выход	40 мА
Флеш-память	32 Кб (ATmega328) при этом 2 Кб используются для загрузчика
ОЗУ	2 Кб (ATmega328)
Тактовая частота	16 МГц
Размеры	1.85 см x 4.2 см



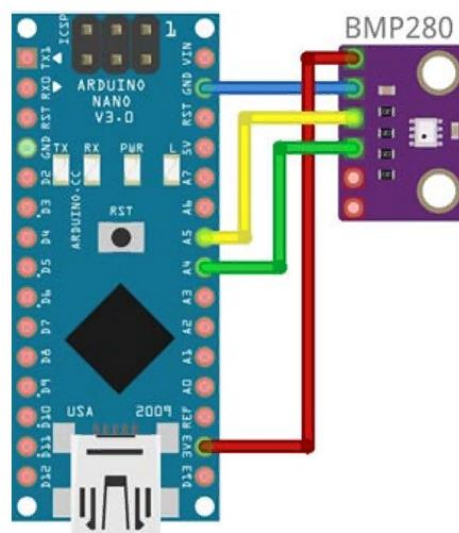
Датчик BME280

- ▶ **VCC, GND** — питание модуля 3.3 В или 5 В
- ▶ **SCL** — линия тактирования (Serial CLock)
- ▶ **SDA** — линия данных (Serial Data)
- ▶ Напряжение питания: 3.3 В – 5 В
- ▶ Рабочий ток: 1 мА
- ▶ Диапазон измерения давления: 300 гПа – 1100 гПа (точность ± 1.0 гПа)
- ▶ Диапазон измерения температуры: -40 °C до $+85$ °C (точность ± 0.5 °C)
- ▶ Диапазон измерения влажности: 20 % до 80 % (точность ± 3 %)
- ▶ Интерфейс: I2C
- ▶ Габариты: 12 мм x 10 мм



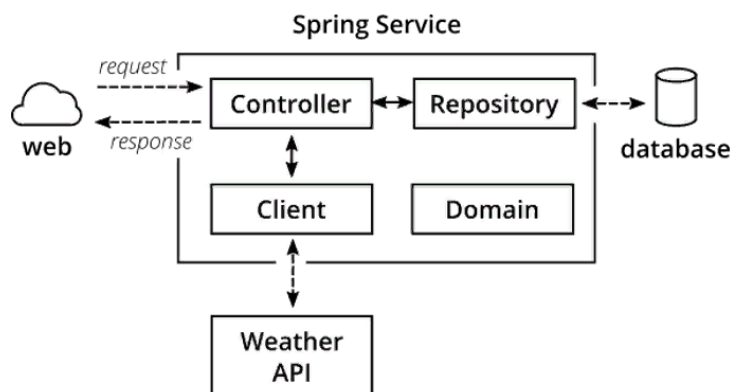
Подключение Датчика к Плате

Для соединения с Ардуино нужно подключить выходы следующим образом: соединить землю с Ардуино и на датчике, VCC – на 3,3В, SCL / SCK – к аналоговому контакту A5, SDA / SDI – к A4.



Изм.	Лист	№ докум.	Подпись	Дата

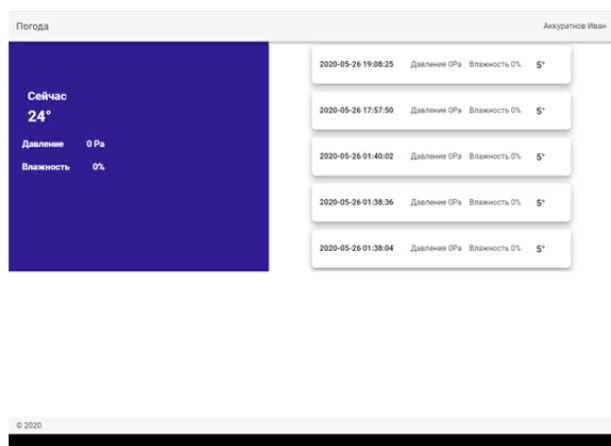
UML Клиент-серверного приложения



UML Запрос в БД



Web-страница



Благодарю за внимание

