**ДИПЛОМНА РАБОТА**

на тема

„Платформа за събиране на данни в реално време”

Дипломант: **Иван Цветомиров Иванов**

Специалност: **Разпределени системи и мобилни технологии**

Факултетен номер: **5MI3400182**

Научен ръководител:

**Проф. д-р Милен Петров**

**Съдържание**

**Глава 1. Увод**

1.1. Актуалност на проблема и мотивация (0,5-1стр.)

1.2. Цел и задачи на дипломната работа (1-2стр.)

1.3. Очаквани ползи от реализацията (1-2стр.)

1.4. Структура на дипломната работа (0,5-1стр.)

**Глава 2. Технически обзор** (10-15стр.)

2.1. Основни дефиниции

2.2. Подходи, методи (евентуално модели и стандарти) за решаване на проблемите

2.3. Съществуващи решения (практически реализации)

2.4. Избор на критерии за сравнение и сравнителен анализ на решения/методи/стандарти/...

2.5. Изводи

**Глава 3. Използвани технологии, платформи и/или методологии** (10-15стр.)

3.1. Изисквания към средствата (технологии, платформи и методологии)

3.2. Видове средства (технологии, платформи и методологии) и начин и място за използването им – сравненителен анализ

3.3. Избор на средствата (технологии, платформи и методологии)

3.4. Изводи

**Глава 4. Анализ**

4.1. Концептуален модел

4.2. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...)

4.3. Качествени (нефункционални) изисквания (като напр. преносимост, използваемост, скалируемост, поддръжка, ...)

4.4. Работни (бизнес) процеси

4.5. Изводи

**Глава 5. Проектиране**

5.1. Обща архитектура – напр. слоеве, модули, блокове, компоненти...

5.2. Модел на данните (напр. база данни, файлова структура, ...)

5.3. Диаграми (на структура и поведение - по слоеве и модули, с извадки от кода)

5.4. Потребителски интерфейс (опционално)

5.5. Ресурсни и спомагателни модули (опционално)

**Глава 6. Реализация, тестване/експерименти и внедряване**

6.1. Реализация на модулите

6.2. Системна интеграция (опционално)

6.3. Планиране на тестването - тестови сценарии, процедури, ...

6.4. Модулно и системно тестване

6.5. Анализ на резултатите от тестването и начин на отразяването им

6.6. Експериментално внедряване (технологични изисквания, инсталиране, условия, използване, ...)

**Глава 7. Заключение**

7.1. Обобщение на изпълнението на началните цели

7.2. Насоки за бъдещо развитие и усъвършенстване

# Глава 1. Увод

# Актуалност на проблема и мотивация

Последните няколко години доста силно навлизат интелигентни сензори за измерване на различни параметри на околната среда, съответно и доста нашумяват различни уеб платформи за събиране, обработка и съхранение на различни измервания в реално време.

Подобни проекти са доста предпочитани от любители на метеорологията и чрез публикуваните данни, може да бъде реализирана мрежа за „отворени данни“, които да се използват за различни цели.

# Цел и задачи на дипломната работа

Основна цел на текущата дипломна работа е проектирането и разработването на платформа за събиране, обработване и съхранение на данни от различни измервателни IoT устройства в реално време.

При проектирането и разработването на проекта, трябва да бъде предвидено крайното предназначение – работа с любители на метрологията и да бъде предоставен удобен и лесен потребителски интерфейс, но също така и лесен, но сигурен начин за публикуване на данни в платформата.

Освен определяне на основни изисквания към платформата, трябва да бъдат проучени и подходящи бази от данни за съхранение на обработените данни.

При разработването на проекта ще бъде основен фокус върху използването на езици, работни рамки и проекти с отворен код.

Също така самият проект ще се разработва като проект с „отворен код“.

# Структура на дипломната работа

Дипломанта работа е структурирана от следните глави. Всяка глави описва конкретна стъпка от анализиране на основните изисквания към проекта, проектирането на основните модули и тяхната разработка с конкретно използвани технологии.

* Глава 1
* Глава 2
* Глава 3
* Глава 4
* Глава 5

# Глава 2. Технически обзор

…

# Основни дефиниции

Основните процеси в проекта ще бъдат основно върху комуникацията между IoT устройствата и платфомата, обработката и анализирането на данните и тяхното съхранение.

Всеки регистриран потребител с потвърден профил ще може да добавя неограничен брой устройства.

# Подходи за решаване на проблемите

….

# Примерни решения

Проекта Sensor Community е реализиран по следния начин:

# Изводи

# Глава 3. Анализ

4.1. Концептуален модел

4.2. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...)

4.3. Качествени (нефункционални) изисквания (като напр. преносимост, използваемост, скалируемост, поддръжка, ...)

4.4. Работни (бизнес) процеси

4.5. Изводи

ТОДО

1. Сигурност
2. Производителност
3. Комуникация между IoT и платформата
4. Управление и анализ на устройства и данни

# Функционални изисквания

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

# Нефункционални изисквания

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

# Основни бизнес процеси

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

# Глава 4. Проектиране

Тук се разглежда и описва процеса на проектиране на отделните модули от платформата основавайки се на основните изисквания и ограничения към тях.

# Обща архитектура

Общата софтуерна архитектура на разработеното приложение е показана на долната фигура.



*Фигура 4: Обща архитектура на приложението*

За софтуерна архитектурна реализация е избран подхода с използването на уеб услуги и разделянето на отделни три слоя, като всеки слой да има определени функции и изпълнява определени роли и изисквания.

1. Слой UI
2. Сървърен слой
3. Слой за съхранение

Frontend – това е клиентката част на приложението, тук е включен и потребителския интерфейс UI, също така той изцяло двупосочно комуникира с сървърната част.

Backend – сървърната част на приложението и тук е реализация на бизнес логиката, в конкретния случай е важен и подбора на софтуерна архитектура, модел за управление на данните в БД и т.н.

Database – отговаря за съхранение на обработените данните от сървърната част.

Избран е подхода за използване на уеб услуги, по този начин имаме разделение между клиентската и сървърната част. За разлика от класическия метод за разработка на уеб приложения, и изгледите да бъдат генерирани от сървърното приложение, то тук имаме следните предимства:

1.

2.

3.

4.

Както беше разгледано има доста предимства избрания принцип за реализация чрез използване на уеб услуги, пред класическия модел за реализация на уеб приложения за директно изпращане на изгледи.

# Реализация на клиентката част (UI)

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

# Реализация на сървърната част (Server)

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

# Модел на данните

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

# Глава 5. Използвани технологии

В тази глава са разгледани изполваните езици, работни рамки и технологии за софтуерната реализация на проекта.

Основни изисквания

Изводи

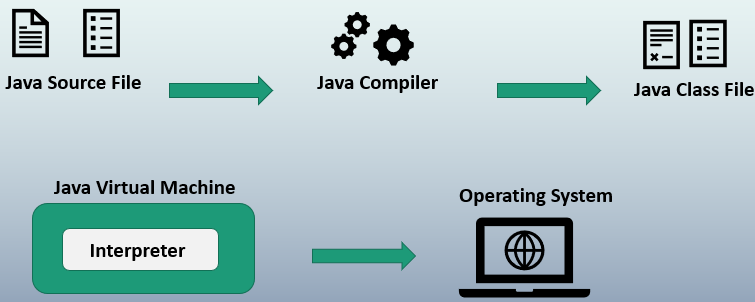
# Java

Java е обектно-ориентиран език за програмиране разработен от Sun, в момента се разработва от Oracle и голяма поддръжка от различни софтуерни компании и ораганизации, като: SAP, VMware, Eclipse, Apache и други.

Изходния код на Java не се компилира директно до машинен код, а до псевдо код, които не се изпълнява директно от операционата систена, а от Java Virtial Machine, което е основното предимство на Java, защото езика е платформено независим и веднъж компилирано приложението, то може да се изпълни на различни устройства без значение от операционната система, единствено изискване е да има инсталиран Java Runtime Edition.

Голямата мобилност на създадените Java приложения прави езика широко използван език за разработка на различен тип проекти, като уеб приложения, уеб услуги, мрежови приложения, десктоп приложения и мобилни приложения за Android операционна система.

Процесът на компилация и изпълнение на Java приложение е показан на фигура 1.



Фигура 1. Java Virtual Machine

Java виртуалната машина при изпълнение на компилираната програма се грижи за работата с паметта и нейното освобождаване от обекти, които не се използват и оптимизиране на процесорно време. Факта, че създадените приложения не работят директно с ресурсите, дава добра степен на сигурност. Но също това е съществен недостатък, защото за самата виртуална машина са необходими допълнителни ресурси като памет и процесорно време за работа.

Съществува вариант за допълнителни оптимизации на използваните ресурси от виртуалната машина и реализираното приложение, такъв проект е GraalVМ.

*Основни Java пакети/модули.*

*Java EE/Jakarta*

# Spring

Spring e широко използван framework с отворен код за разработка на Java приложения (уеб приложения и услуги, автоматизации на бизнес процеси и работа с бази от данни). В голяма степен надгражда Java EE/Jakarta и предлага широк спектър от оптимизации и автоматизации при реализацията на основните бизнес процеси, работа с бази от данни, допълнителни библиотеки и услуги от трети страни.

1. **Spring Framework**

Реализация на високо ниво за разработка на уеб приложения чрез Java EE/Jakarta и в голяма степен е алтернатива на Enterprise Java Beans. На фигура 2 е показана основната структура на приложната рамка за разработка.



Фигура 2. Архитектура на Spring Framework

Чрез Spring Framework могат да се реализират уеб приложения чрез MVC шаблона и REST базирани уеб услуги.

Към момента на разработване на проекта, актуалната версия на Spring Framework е 6.0.0 която е Jakarta базирана и подържа следните Servlet имплементации:

* Apache Tomcat xxx
* Eclipse Jetty xxx

*IoC*

1. **Spring Boot**

Spring Boot предлага готова конфигурация на Spring Framework базирана на шаблона „Convention over configuration“, включително предлага и вграден приложен уеб сървър – Apache Tomcat, Eclipse Jetty и Netty. Архитектура на Spring Boot приложение e показана на фигура 3.



Фигура 3.

*Spring Web*

*Spring Webflux*

1. **Spring Security**

Тук се предлагат голям набор от класове и конфигурации за authentication и authorization на разработеното приложение. Принципът на работа на Spring Security e показан на фигура 4.



Фигура 4.

Spring Resource Server

Spring Authorization Server

1. **Spring Data**

….

# Mongo Db

# Други

Използването на Apache Kafka за messaging платформа предлага редица предимства, включително

* **Висока производителност:** Kafka е проектирана да бъде много производителна и може да обработва големи обеми данни в реално време.
* **Гъвкавост:** Kafka може да се използва за различни приложения, включително обработка на събития, анализ на данни и интеграция на системи.
* **Скейлируемост:** Kafka е мащабируема система, която може да се разширява, за да поддържа нарастващия обем на данни.
* **Надеждност:** Kafka е надеждна система, която е проектирана да може да възстановява данните в случай на срив.
* **Отвореност:** Kafka е отворен код, което означава, че може да бъде разглеждана и променяна от всеки.

Като цяло, Apache Kafka е мощна и гъвкава messaging платформа, която може да се използва за различни приложения. Ако търсите високопроизводителна, мащабируема и надеждна система за обмен на данни, Kafka е чудесен избор.

Ето някои конкретни примери за приложения, за които Kafka може да се използва:

* **Обработка на събития:** Kafka може да се използва за събиране и обработка на събития от различни източници в реално време. Това може да се използва за подобряване на производителността, видимостта и вземането на решения.
* **Анализ на данни:** Kafka може да се използва за съхраняване и анализ на големи обеми данни от различни източници. Това може да се използва за идентифициране на тенденции, модели и аномалии.
* **Интеграция на системи:** Kafka може да се използва за интегриране на различни системи и приложения. Това може да се използва за подобряване на ефективността и гъвкавостта.

Ако търсите messaging платформа, която може да се използва за различни приложения, Apache Kafka е чудесен избор.

* TypeScript
* React
* Material UI

# Глава 6. Софтуерна реализация

**Глава 6. Реализация, тестване/експерименти и (евентуално) внедряване** (10-15стр.)

6.1. Реализация на модулите

6.2. Системна интеграция (опционално)

6.3. Планиране на тестването - тестови сценарии, процедури, ...

6.4. Модулно и системно тестване

6.5. Анализ на резултатите от тестването и начин на отразяването им

6.6. Експериментално внедряване (технологични изисквания, инсталиране, условия, използване, ...)

# Глава 7. Заключение

7.1. Обобщение на изпълнението на началните цели

7.2. Насоки за бъдещо развитие и усъвършенстване

# Използвана литература

1. Източник 1
2. Източник 2
3. Източник 3
4. Източник 4
5. …

# Приложение № 1: Терминологичен речник

# Приложение № 2: Потребителски наръчник

**Приложения (опционално)**

**Приложение 1: Терминологичен речник (опционално)** (1-2стр.)

**Приложение 2, 3, ....:** напр. код или извадки от кода, наръчник на потребителя, екрани от потребителския интерфейс, спецификации, диаграми, и др.

**Изисквания за оформяне на дипломната работа:**

1. Това е препоръчителен шаблон, в зависимост от конкретното задание.
2. Йерархията на структуриране на съдържанието да не бъде повече от 3 нива, номерирани с арабски цифри – напр. 1.2.3.
3. Чуждестранните термини да бъдат преведени, а където това не е възможно – цитирани в *курсив* и нечленувани.
4. Страниците да бъдат номерирани с арабски цифри, в долния десен ъгъл.
5. Използваният шрифт за основния текст на описанието да бъде Times 12 или Arial 10, и Courier 9 за кода, с междуредие 16pt.
6. Да се избягват пренасянията на нова страница на заглавия на секции, фигури и таблици.
7. Да се избягват празни участъци на страници вследствие пренасянето на фигури на нова страница.
8. Всички фигури и таблици да бъдат номерирани и именовани (непосредствено след фигурата или таблицата).
9. Всички фигури и таблици да бъдат цитирани в текста.
10. Използваните фигури от други източници да бъдат цитирани.
11. Всички цитати да бъдат отразени в списъка на използваната литература.
12. Всички източници от списъка на използваната литература да бъдат цитирани в текста.
13. Използваната литература да се цитира съгласно MLA Style - <http://www.library.mun.ca/guides/howto/mla.php>