|  |  |
| --- | --- |
| |  | | --- | | **УНИВЕРЗИТЕТ У НОВОМ САДУ**  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  **НОВИ САД**  **Департман за рачунарство и аутоматику**  **Одсек за рачунарску технику и рачунарске комуникације**  **ЗАВРШНИ (BACHELOR) РАД**  **Кандидат:**  **Број индекса:**  **Тема рада:**  **Ментор рада:**  **Нови Сад, месец, година** | |

|  |  |  |  |
| --- | --- | --- | --- |
| Редни број, **РБР**: | |  | |
| Идентификациони број, **ИБР**: | |  | |
| Тип документације, **ТД**: | | Монографска документација | |
| Тип записа, **ТЗ**: | | Текстуални штампани материјал | |
| Врста рада, **ВР**: | | Завршни (Bachelor) рад | |
| Аутор, **АУ**: | |  | |
| Ментор, **МН**: | |  | |
| Наслов рада, **НР**: | |  | |
| Језик публикације, **ЈП**: | | Српски / латиница | |
| Језик извода, **ЈИ**: | | Српски | |
| Земља публиковања, **ЗП**: | | Република Србија | |
| Уже географско подручје, **УГП**: | | Војводина | |
| Година, **ГО**: | |  | |
| Издавач, **ИЗ**: | | Ауторски репринт | |
| Место и адреса, **МА**: | | Нови Сад; трг Доситеја Обрадовића 6 | |
| Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога) | |  | |
| Научна област, **НО**: | | Електротехника и рачунарство | |
| Научна дисциплина, **НД**: | | Рачунарска техника | |
| Предметна одредница/Кqучне речи, **ПО**: | |  | |
| **УДК** | |  | |
| Чува се, **ЧУ**: | | У библиотеци Факултета техничких наука, Нови Сад | |
| Важна напомена, **ВН**: | |  | |
| Извод, **ИЗ**: | |  | |
| Датум прихватања теме, **ДП**: | |  | |
| Датум одбране, **ДО**: | |  | |
| Чланови комисије, **КО**: | Председник: |  |
|  | Члан: |  | Потпис ментора |
|  | Члан, ментор: |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Accession number, **ANO**: | |  | |
| Identification number, **INO**: | |  | |
| Document type, **DT**: | | Monographic publication | |
| Type of record, **TR**: | | Textual printed material | |
| Contents code, **CC**: | | Bachelor Thesis | |
| Author, **AU**: | |  | |
| Mentor, **MN**: | |  | |
| Title, **TI**: | |  | |
| Language of text, **LT**: | | Serbian | |
| Language of abstract, **LA**: | | Serbian | |
| Country of publication, **CP**: | | Republic of Serbia | |
| Locality of publication, **LP**: | | Vojvodina | |
| Publication year, **PY**: | |  | |
| Publisher, **PB**: | | Author’s reprint | |
| Publication place, **PP**: | | Novi Sad, Dositeja Obradovica sq. 6 | |
| Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes) | |  | |
| Scientific field, **SF**: | | Electrical Engineering | |
| Scientific discipline, **SD**: | | Computer Engineering, Engineering of Computer Based Systems | |
| Subject/Key words, **S**/**KW**: | |  | |
| **UC** | |  | |
| Holding data, **HD**: | | The Library of Faculty of Technical Sciences, Novi Sad, Serbia | |
| Note, **N**: | |  | |
| Abstract, **AB**: | |  | |
| Accepted by the Scientific Board on, **ASB**: | |  | |
| Defended on, **DE**: | |  | |
| Defended Board, **DB**: | President: |  |
|  | Member: |  | Menthor's sign |
|  | Member, Mentor: |  |  |

**Захвалност**

**Садржај**

[1. Увод **Error! Bookmark not defined.**](#_Toc296447924)

[2. Теоријске основе **Error! Bookmark not defined.**](#_Toc296447925)

[2.1 . **Error! Bookmark not defined.**](#_Toc296447926)

[2.2 . **Error! Bookmark not defined.**](#_Toc296447927)

[3. Концепт решења **Error! Bookmark not defined.**](#_Toc296447928)

[3.1 . **Error! Bookmark not defined.**](#_Toc296447929)

[3.1.1 . **Error! Bookmark not defined.**](#_Toc296447930)

[3.1.2 . **Error! Bookmark not defined.**](#_Toc296447931)

[3.1.3 . **Error! Bookmark not defined.**](#_Toc296447932)

[4. Програмско решење **Error! Bookmark not defined.**](#_Toc296447935)

[4.1 . **Error! Bookmark not defined.**](#_Toc296447936)

[4.1.1 . **Error! Bookmark not defined.**](#_Toc296447937)

[4.1.2 . **Error! Bookmark not defined.**](#_Toc296447938)

[4.2 . **Error! Bookmark not defined.**](#_Toc296447939)

[4.2.1 . **Error! Bookmark not defined.**](#_Toc296447940)

[4.3 . **Error! Bookmark not defined.**](#_Toc296447942)

[4.3.1 . **Error! Bookmark not defined.**](#_Toc296447943)

[4.4 . **Error! Bookmark not defined.**](#_Toc296447944)

[4.4.1 K. **Error! Bookmark not defined.**](#_Toc296447945)

[5. Резултати **Error! Bookmark not defined.**](#_Toc296447946)

[6. Закључак **Error! Bookmark not defined.**](#_Toc296447947)

[7. Литература **Error! Bookmark not defined.**](#_Toc296447948)

**Списак слика**

*Слика 1. .* **Error! Bookmark not defined.**

*Слика 2. .* **Error! Bookmark not defined.**

*Слика 3. .* **Error! Bookmark not defined.**

*Слика 4. .* **Error! Bookmark not defined.**

**Списак табела**

*Табела 1. .* **Error! Bookmark not defined.**

*Табела 2. .* **Error! Bookmark not defined.**

**Скраћенице**

# Увод

Тема проучавања овог дипломског рада фокусирана је на базама података, са конкретним акцентом на *Time Series Database*.

У књизи “*Time Series Databases New Ways to Store and Access Data”* аутора *Ted Dunning* и *Ellen Friedman* прва мантра је представљена реченицом *“Collect your data as if your life depends on it!”.* Ова реченица јасно указује на вриједност података и важност исправног чувања истих.

У овом раду искориштена је за систематично и ефикасно сакупљање, чување и надгледање података добијених путем сензора распоређених у канцеларијама пословне зграде. Три кључна параметра - температура, влажност и ниво угљен моноксида - пажљиво се прате у реалном времену, пружајући особљу одржавања детаљан увид у околности унутар објекта. Свака канцеларија опремљена је са по три сензора, чинећи систем свеобухватним и прецизним у детекцији потенцијалних проблема.

Једна од кључних функционалности овог система је визуализација података путем графикона, што омогућава радницима да лако прате параметре на нивоу зграде и сваке појединачне канцеларије. Ова транспарентност и брза доступност информација пружају особљу одржавања могућност да предузму одговарајуће кораке у реалном времену, и тиме драстично смањује вријеме реакције одговорних лица на опасну ситуацију.

Нарочито значајно, систем не само да чува податке, већ и активно реагује на неочекиване вриједности мјерења. Уколико измјерене вриједности изађу из опсега дефинисаних нормалних вриједности, систем аутоматски обавјештава особље одржавања. У случају екстремних вриједности, када постоји пријетња по сигурност, радници су непосредно обавјештени о потреби евакуације, чиме се обезбјеђује брза и ефикасна реакција на потенцијалне опасности.

У наредним одјељцима овог рада, размотрићемо концепт рјешења, теоријске основе технологија коришћених у имплементацији као и детаље имплементације система.

# Опис реалног система

У овом делу се истражује реални систем, описује начин његовог функционисања кроз идентификована правила пословања, као и захтеви који су постављени пред систем током фазе спецификације корисничких захтева.

## Класе корисника и њихове карактеристике

Систем разликује три групе корисника:

1. администратор
2. радник одржавања система
3. радник

Корисници типа администратор су овлаштена лица која имају право приступа свим функционалностима система. Администратор је задужен да стање у реалном систему евидентира у информациони систем, односно да у систему региструје сваку канцеларију из пословне зграде, забиљежи сваки инсталиран сензор и слично. Све у циљу да информациони систем вјеродостојно осликава реални, и омогући олакшавање надгледања очитаних вриједности са сеснзора као и аутоматизацију система за слање обавјештења.

Корисници типа радник одржавања система су људи који су задужени за надгледање очитавања са сензора и којима је омогућено слање наређења о евакуацији свим радницима у згради. Због вјеродостојности информација и стабилности система радник одржавања система нема право приступа административном дијелу софтвера, тачније функционалности 2.2.7 из поглавља Функционални захтјеви.

Корисници типа радник представљају запослене у пословном објекту који се надгледа. Они имају најограниченија права приступа, која се своде на посмарање вријености очитаних параметара из њихове канцеларије.

## Функционални захтјеви

### *Пријава на систем*

Корисници који су претходно регистровани у систем од стране администратора могу да се пријаве на ситем користећи свој *email* и лозинку.

Корисник типа радник може да приступи функционалностима 2.2.3, 2.2.4 и 2.2.5.

Корисник типа радник одржавања система може да приступи функционалностима 2.2.2, 2.2.3, 2.2.4, 2.2.5, 2.2.6.

Корисник типа администратор може да приступ свим функционалностима система.

### *Симулација уласка/изласка радника из пословног објекта*

Систем треба да омогући кориснику избор пословног простора и симулацију уласка или изласка запосленог. Корисник би требао да може да одабере конкретног запосленог, а систем би затим требало да генерише сценарио уласка или изласка. Ова функционалност такође би требало да обезбеди логичку проверу, као што је ограничење да корисник може да симулира излазак само за оне запослене који се у том тренутку налазе унутар пословног објекта. Додатно, систем би требало да чува податке о симулираним уласцима и излазима, омогућавајући каснију анализу и праћење активности запослених.

### *Надгледање параметара на нивоу пословног објекта*

Систем треба да омогући кориснику систематичан графички приказ свих параметара (угљен моноксида, температуре и влажности ваздуха) на нивоу пословног објекта. Графикон треба да приказује вриједности параметара у реалном времену.

### *Надгледање параметара на нивоу канцеларије*

Систем треба да омогући кориснику систематичан графички приказ свих параметара (угљен моноксида, температуре и влажности ваздуха) на нивоу сваке канцеларије у пословном објекту. Графикон треба да приказује вриједности параметара у реалном времену. Такође, потребно је омогућити кориснику одабир параметра који жели да прикаже на графикону за одабрану канцеларију.

### *Систем за провјеру безбједности*

Потребно је имплементирати систем за провјеру безбиједности. Овај систем подразумијева константу провјеру сачуваних вриједности мјерења сензора, упоређивање са очекиваним вриједностима и обавјештавање одговорних лица уколико вриједности изађу из очекиваног опсега.

Слање обавјештења се врши на два начина:

* Обавјештење на корисничком интерфејсу

Уколико дође до било каквог одступања од опсега очекиваних вриједности на неком од сензора потребно је информисати раднике одржавања пословног објекта путем обавјештења на корисничком интерфејсу.

* Обавјештење путем електронске поште

Уколико се на неком од сензора забиљежи нижа вриједност од очекиване послаће се обавјештење запосленим лицима из канцеларије у којој је измјерена неочекивана вриједност путем електронске поште, али само онима од њих који се тренутно налазе у пословном објекту.

Уколико се на неком од сензора забиљежи виша вриједност од очекиване послаће се упозорење, путем електронске поште свим запосленим лицима који се тренутно налазе у пословном објекту.

### *Слање упозорења свим запосленим*

Поред аутоматског слања обавјештења које врши систем по правилима дефинисаним у претходном функционалном захтјеву, потребно је омогућити раднику из одржавања који надгледа све параметре у згради да кликом на дугме пошаље свим запосленима наређење о евакуацији уколико примијети необична очитавања.

У том случају свим запосленим унутар зграде се шаље електронска пошта којом се запослени обавјештавају да хитно напусте зграду.

### *Читање, креирање, брисање и модификовање података*

Потребно је омогућити приказ, брисање, модификовање и креирање нових података о канцеларијама, сензорима, типовима сензора, инсталацији сензора и радницима.

# Опис модела података

Ово поглавље детаљно разлаже модел података на којем се темељи информациони систем. У имплементацији софтвера кориштене су две различите базе података. Релациона база података је коришћена за чување података о свим ентитетима релевантним за систем, као што су зграде, канцеларије, радници, сензори итд. Са друге стране, нерелациона база података, тачније база података временских серија је коришћена за чување свих мjерења са сензора, као и догађаја радника који улазе/излазе из зграде. Овакав приступ омогућава оптимално управљање различитим врстама података у складу са њиховим специфичностима и потребама система.

## Релациона база података

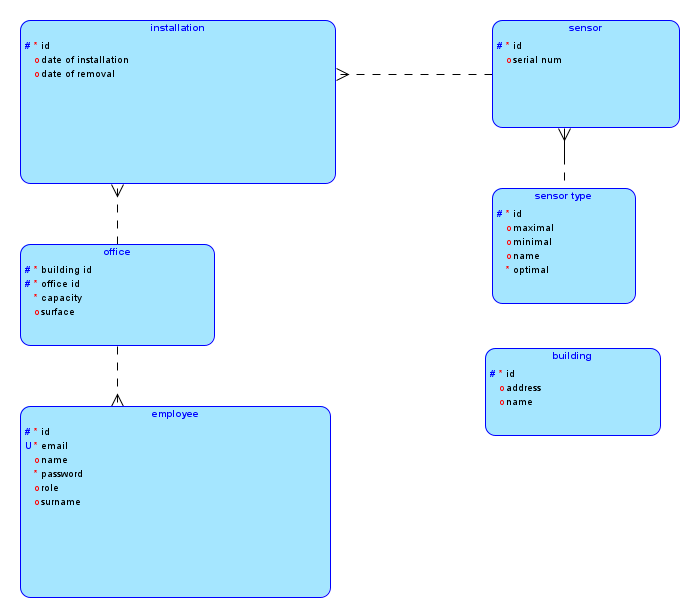
У оквиру система кориштена је релациона база података за чување и руковање са информација о овим ентитетима у систему:

* зграда
* радник
* тип радника
* инсталација
* канцеларија
* улога
* сензор
* име сензора
* тип сензора

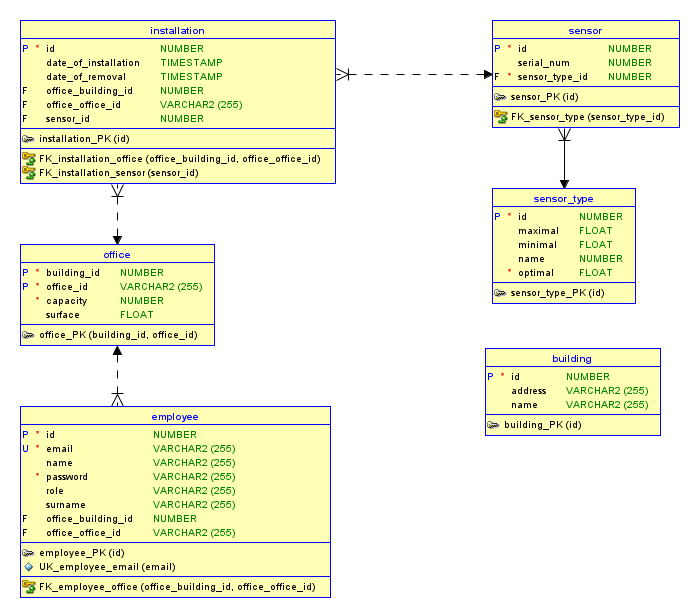
За поједностављено дефинисање шеме базе података, примењен је *Java Persistence API (JPA)* унутар *Spring Boot* оквира. Овај приступ ефикасно генерише потребну шему базе података на основу модела коришћеног у *Java* апликацији. Важно је нагласити да изостанак потребе за ручном креацијом детаљне шеме потиче из њене нискe комплексности. Структура базе података овог система лако и идентично може бити креирана путем *ЈPA*, обезбеђујући при том конзистентност и усклађеност са захтевима апликације. Овакав приступ не само да смањује вријеме израде, већ такође осигурава идеално окружење за рад са подацима у систему.

У наредном поглављу представљена је логичка и имплементациона шема базе података. Шеме су генерисане кориштењем алата *Oracle Data Modeler* од *ddl* скрипте коју је генерисао *ЈРА*. Овај корак има за циљ пружање визуелне репрезентације и детаљног објашњења шеме базе података која је основа система, доприносећи бољем разумевању архитектуре података у оквиру система.

### *Модел логичке шеме базе података*



### *Модел релационе шеме базе података*



### *Опис имплементационог модела базе података*

У овом поглављу представљене су спецификације свих релација, приказаних у табеларном облику са свим њиховим карактеристикама и примењеним ограничењима. Ова детаљна анализа има за циљ да пружи потпуни увид у структуру података и међусобне везе унутар система.

#### Зграда

Шема релације Зграда садржи основне податке о згради.

|  |  |  |  |
| --- | --- | --- | --- |
| **Колоне у табели** | | | |
| **Назив колоне** | **Тип података** | ***Nullable*** | **Опис колоне** |
| *id* | *BIGSERIAL* | Не | Јединствени идентификатор |
| *address* | *VARCHAR* (255) | Да | Адреса зграде |
| *name* | *VARCHAR* (255) | Да | Назив зграде |

|  |  |
| --- | --- |
| **Ограничење примарног кључа** | |
| **Назив ограничења** | **Колоне** |
| *PK*\_ЗГРАДА | *id* |

#### Радник

Шема релације Радник садржи основне податке о раднику.

|  |  |  |  |
| --- | --- | --- | --- |
| **Колоне у табели** | | | |
| **Назив колоне** | **Тип података** | ***Nullable*** | **Опис колоне** |
| *id* | *BIGSERIAL* | Не | Јединствени идентификатор |
| *email* | *VARCHAR(255)* | Не | *Еmail* адреса радника |
| *name* | *VARCHAR(255)* | Не | Име |
| *surname* | *VARCHAR(255)* | Не | Презиме |
| *password* | *VARCHAR(255)* | Не | Лозинка |
| *role* | *VARCHAR(255)* | Не | Улога у систему |
| office\_id | *VARCHAR(255)* | Не | Јединствени идентификатор канцеларије у којој ради радник |

|  |  |
| --- | --- |
| **Ограничење примарног кључа** | |
| **Назив ограничења** | **Колоне** |
| *PK*\_РАДНИКА | *id* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Ограничење референцијалног интегритета** | | | |
| **Назив ограничења** | **Колоне** | **Референцирана табела** | **Референциране колонe** |
| *FK*\_КАНЦЕЛАРИЈА\_ЗГРАДА | office\_building\_id | Канцеларија | *building\_id* |
| *FK*\_КАНЦЕЛАРИЈА\_КАНЦЕЛАРИЈА | office\_office\_id | Канцеларија | *office\_id* |

|  |  |
| --- | --- |
| **Ограничење јединствености** | |
| **Назив ограничења** | **Колоне** |
| *UQ*\_еmail | *email* |

#### Канцеларија

Шема релације Канцеларија садржи основне податке о канцеларији.

|  |  |  |  |
| --- | --- | --- | --- |
| **Колоне у табели** | | | |
| **Назив колоне** | **Тип података** | ***Nullable*** | **Опис колоне** |
| *building\_id* | *BIGINT* | Не | Јединствени идентификатор зграде у којој се налази канцеларија |
| *office\_id* | *VARCHAR(255)* | Не | Јединствени идентификатор канцеларије |
| *capacity* | *INTEGER* | Не | Број радних станица у канцеларији |
| *surface* | *FLOAT(53)* | Не | Површина канцеларије у метрима квадратним |

|  |  |
| --- | --- |
| **Ограничење примарног кључа** | |
| **Назив ограничења** | **Колоне** |
| *PK*\_КАНЦЕЛАРИЈА | *building\_id+ office\_id* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Ограничење референцијалног интегритета** | | | |
| **Назив ограничења** | **Колоне** | **Референцирана табела** | **Референциране колонe** |
| *FK*\_ЗГРАДА | *building\_id* | Зграда | *id* |

#### Сензор

Шема релације Сензор садржи основне податке о сензопу.

|  |  |  |  |
| --- | --- | --- | --- |
| **Колоне у табели** | | | |
| **Назив колоне** | **Тип података** | ***Nullable*** | **Опис колоне** |
| *id* | *BIGSERIAL* | Не | Јединствени идентификатор |
| *serial\_num* | *BIGINT* | Не | Серијски број сензора |
| *sensor\_type\_id* | *BIGINT* | Не | Јединствени идентификатор типа сензора којем припада сензор |

|  |  |
| --- | --- |
| **Ограничење примарног кључа** | |
| **Назив ограничења** | **Колоне** |
| *PK*\_СЕНЗОР | *id* |

|  |  |  |  |
| --- | --- | --- | --- |
| **Ограничење референцијалног интегритета** | | | |
| **Назив ограничења** | **Колоне** | **Референцирана табела** | **Референциране колонe** |
| *FK*\_СЕНЗОР\_ТИП | *sensor\_type\_id* | Тип сензора | *id* |

#### Тип сензора

Шема релације Тип сензора садржи основне податке о типу сензора.

|  |  |  |  |
| --- | --- | --- | --- |
| **Колоне у табели** | | | |
| **Назив колоне** | **Тип података** | ***Nullable*** | **Опис колоне** |
| *id* | *BIGSERIAL* | Не | Јединствени идентификатор |
| *maximal* | *FLOAT(53)* | Не | Максимална сигурна вриједност |
| *minimal* | *FLOAT(53)* | Не | Минимална сигурна вриједност |
| *name* | *SMALLINT* | Не | Назив типа сензора |
| *optimal* | *FLOAT(53)* | Не | Оптимална вриједност |

|  |  |
| --- | --- |
| **Ограничење примарног кључа** | |
| **Назив ограничења** | **Колоне** |
| *PK*\_ТИП\_СЕНЗОРА | *id* |

#### Инсталација

Шема релације Инсталација садржи основне податке о инсталацији одређеног сензора.

|  |  |  |  |
| --- | --- | --- | --- |
| **Колоне у табели** | | | |
| **Назив колоне** | **Тип података** | ***Nullable*** | **Опис колоне** |
| *id* | *BIGSERIAL* | Не | Јединствени идентификатор |
| *date\_of\_installation* | *TIMESTAMP(6)* | Не | Датум и вријеме инсталације сензора |
| *date\_of\_removal* | *TIMESTAMP(6)* | Да | Датум и вријеме уклањања сензора |
| *office\_building\_id* | *BIGINT* | Не | Јединствени идентификатор зграде у којој је инсталиран сензор |
| *office\_office\_id* | *VARCHAR(255)* | Не | Јединствени идентификатор канцеларије у којој је инсталиран сензор |
| *sensor\_id* | *BIGINT* | Не | Јединствени идентификатор сензора |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ограничење примарног кључа** | | | | |
| **Назив ограничења** | | **Колоне** | | |
| *PK*\_ИНСТАЛАЦИЈА | | *id* | | |
| **Ограничење референцијалног интегритета** | | | | | |
| **Назив ограничења** | **Колоне** | | **Референцирана табела** | **Референциране колонe** | |
| *FK*\_КАНЦЕЛАРИЈА\_ЗГРАДА | office\_building\_id | | Канцеларија | *building\_id* | |
| *FK*\_КАНЦЕЛАРИЈА\_КАНЦЕЛАРИЈА | office\_office\_id | | Канцеларија | *office\_id* | |
| *FK*\_СЕНЗОР | *sensor\_id* | | Сензор | *id* | |

## База података временских серија

## Временске серије представљају мjерења или догађаје који се систематски прате, надгледају и агрегирају током одређеног временског периода. Ови подаци могу да обухватају различите врсте аналитичких информација, укључујући метрике сервера, перформансе апликација, мрежне податке, сензорске податке, догађаје, кликове, трансакције на тржишту и многе друге. Кључна карактеристика која издваја временске серије од обичних података је стална потреба за постављањем питања о њима током времена. Идентификација скупа података као временских серија једноставно се спроводи провером да ли једна од оса представља временски континуум.

## *InfluxDB* пружа корисницима могућност прикупљања, складиштења, упита, обраде и визуализације сирових података високе прецизности, као и агрегираних и узоркованих података. Платформа *InfluxDB* организује временске серије у структуираном формату, где се највишем нивоу налази назив мерења. Затим следи скуп парова кључ/вредност, познатих као тагови, који описују метаподатке, и кључеви/вредности стварних вредности, познатих као поља. Вредности поља у *InfluxDB* могу бити у *облику boolean, int64, float64* или *string*. На крају, сваком скупу вредности придружен је временски печат. Сви подаци се упитују према мерењу, таговима, пољима и временском опсегу.[1]

Основни концепти у *InfluxDB* бази података су[2]:

* **Поља** представљају обавезну компоненту структуре података. Без присуства поља, подаци не могу бити смештени у InfluxDB. Битно је напоменути да поља нису индексирана, што значи да упити који користе вредности поља као филтере морају прегледати све вредности које се подударају са осталим условима у упиту, што може довести до смањења перформанси. Уопштено, поља не би требала садржавати често претраживане метаподатке.
* **Тагови** су опционални, али препоручљиви део структуре података у InfluxDB-у. За разлику од поља, тагови су индексирани, што резултује бржим упитима на таговима, а тагови су идеални за чување често претраживаних метаподатака. Тагови се састоје од кључева и вредности, где су кључеви и вредности чувани као стрингови.
* **Мјерење** делује као контејнер за тагове, поља и временску колону. Име мерења је опис података који се чувају у повезаним пољима. Концептуално, мерење је слично табели у SQL бази података. Једно мерење може припадати различитим политикама задржавања *(retention policies)*.
* **Контеинер** (*Bucket*) је именовано место где се складиште подаци временских серија. Сваки контеинер има политику задржавања, тј. временски период током ког свака тачка података траје.
* **Политика задржавања** *(Retention Policies)* описују колико дуго *InfluxDB* чува податке и колико копија тих података је смештено у кластеру.
* **Тачка** представља појединачни запис који има четири компоненте: мјерење, скуп тагова, скуп поља и временски печат. Свака тачка је јединствено идентификована својом серијом и временским печатом.
* **База података** чува све ове компоненте. *InfluxDB* база података служи као логички контејнер за кориснике, политике задржавања, континуалне упите и временске серије податка. *InfluxDB* не захтијева стикту шему база података, што значи да је једноставно додавати нова мерења, тагове и поља у било које време, олакшавајући рад са временским серијама података.

У овом информационом систему је употребљено два контејнера (bucket).

Први контејнер, *airSensor-RT*, служи за чување података о мјерењима са сензора.

Подаци се у овај контејнер смјештају периодичним извршавањем task-а "Collect air sensor data - real Time Simulation". Помоћу овог упита, подаци се додају са званичног *API* *InfluxDB* и чувају у базу:

import "influxdata/influxdb/sample"

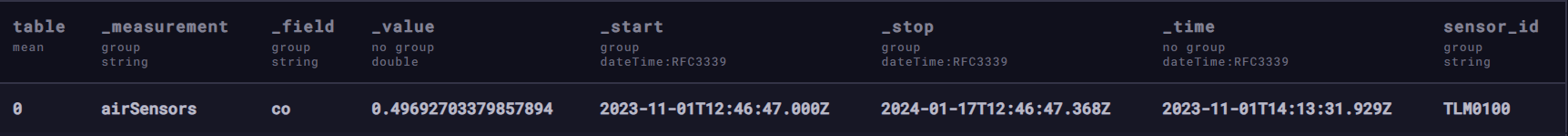
import "date"

sample.data(set: "airSensor")

|> map(fn: (r) => ({r with \_time: date.add(d: 15m, to: r.\_time)}))

|> to(bucket: "airSensor-RT")

У пољу овог контејнера се налазе информације о очитаним вриједностима сензора, док су у тагу смештени бројеви канцеларија у којима су сензори постављени.

 Слика 1. Примјер једне тачке (*point*) у *airSensor-RT* контеинеру.

Други контејнер, *buildingEvents*, служи за складиштење догађаја усласка/изласка радника из пословне зграде. Сваки пут када се симулира улазак или излазак радника из зграде у овај контеинер се сачува нова тачка са потребним инфомрацијама. У пољу овог контејнера се налази информација о правцу кретања радника при уласку или изласку из зграде, док су у тагу смештени идентификациони бројеви радника. Ова структура података је дизајнирана с обзиром на честе упите система, који траже информације о послједњој активности сваког радника. Оптимизација упита је постигнута употребом тага за чување идентификације радника. Пример тога је следећи упит:

from(bucket: \"" + influxDBConnection.getBucket() + "\")

|> range(start: -24h)

|> sort(columns: [\"\_time\"], desc: false)

|> group(columns: [\"employeeId\"])

|> last()

|> filter(fn: (r) => r.\_value == \"in\")

 Слика 1. Примјер једне тачке (*point*) у *buildingEvents* контеинеру.

## Анализа одабира базе у зависности од контекста података

Одабир оптималне базе података за развој апликације представља важну одлуку која значајно утиче на перформансе, скалабилност и одржавање система. Потребно је систематски разматрити више клучних аспеката како би се донела информисана одлука у складу са специфичностима пројекта.

У почетку, кључно је сагледати структуру података која ће се чувати у систему. Релационе базе података пружају чврст и јасно дефинисан модел путем табела, док *NoSQL* базе омогућавају већу флексибилност у приступу подацима. Одлука о моделу података требала би одражавати специфичне потребе апликације и комплексност информација које ће се чувати.

Када је ријеч о перформансама и скалабилности, треба размотрити очекиване захтеве и будући раст апликације. *NoSQL* базе података, посебно оне које подржавају хоризонтално скалирање, често пружају боље перформансе у условима повећаног оптерећења у односу на традиционалне релационе базе, које се често ослањају на вертикално скалирање.

Питање трансакција и доследности података такође је кључно. Ако је неопходна потпуна доследност, релационе базе које подржавају *ACID* трансакције могу бити оптималан избор. Међутим, у ситуацијама где је брзина извршавања приоритет, *NoSQL* базе података које толеришу одређени компромис у доследности могу бити бољи избор.

Различите врсте упита које ће апликација често извршавати такође треба узети у обзир приликом одлучивања између релационих и NoSQL база података. Релационе базе показују добре перформансе у сложеним упитима, док NoSQL базе омогућавају флексибилност у раду са обимним подацима кроз једноставније упите.

Коришћење релационе базе података за ентитете наведене у поглављу 3.1, као што су зграде, канцеларије и радници, у развијаном информационом систему образложава се њиховом природном структуром и захтевима за обраду података. Релациона база података идеално се уклађује са прецизно структурираним подацима ових ентитета који учествују у основним операцијама читања, брисања, додавања и ажурирања. Овакав избор обезбеђује конзистентност, трајност и изолованост података, што је неопходно за стабилан и ефикасан информациони систем.

Такође за овакве ентитете се не очекује уношење екстремних количине података и стога се предвиђа да ће систем управљања базом података успешно обрађивати ове податке, обезбјеђујући при том добре перформансе система.

Са друге стране за чување података о мјерењима сензора и догађајима уласка/изласка радника из пословне зграде изабрана је база података са временским серијама.

Кључно питање које ће јасно указати на потребу за коришћењем базе података с временским серијама јесте: Да ли је у интересу система чување историјске промене вриједности или искључиво тренутних вриједности?

За податке о мјерењима сензора јасно је да је од интереса да се трајно сачува промјена очитаних вриједности у времену. У овом контексту се не ради измјена постојећих вријесноти у бази, него се очитавања у времену чувају, акумулирају и касније обрађују.

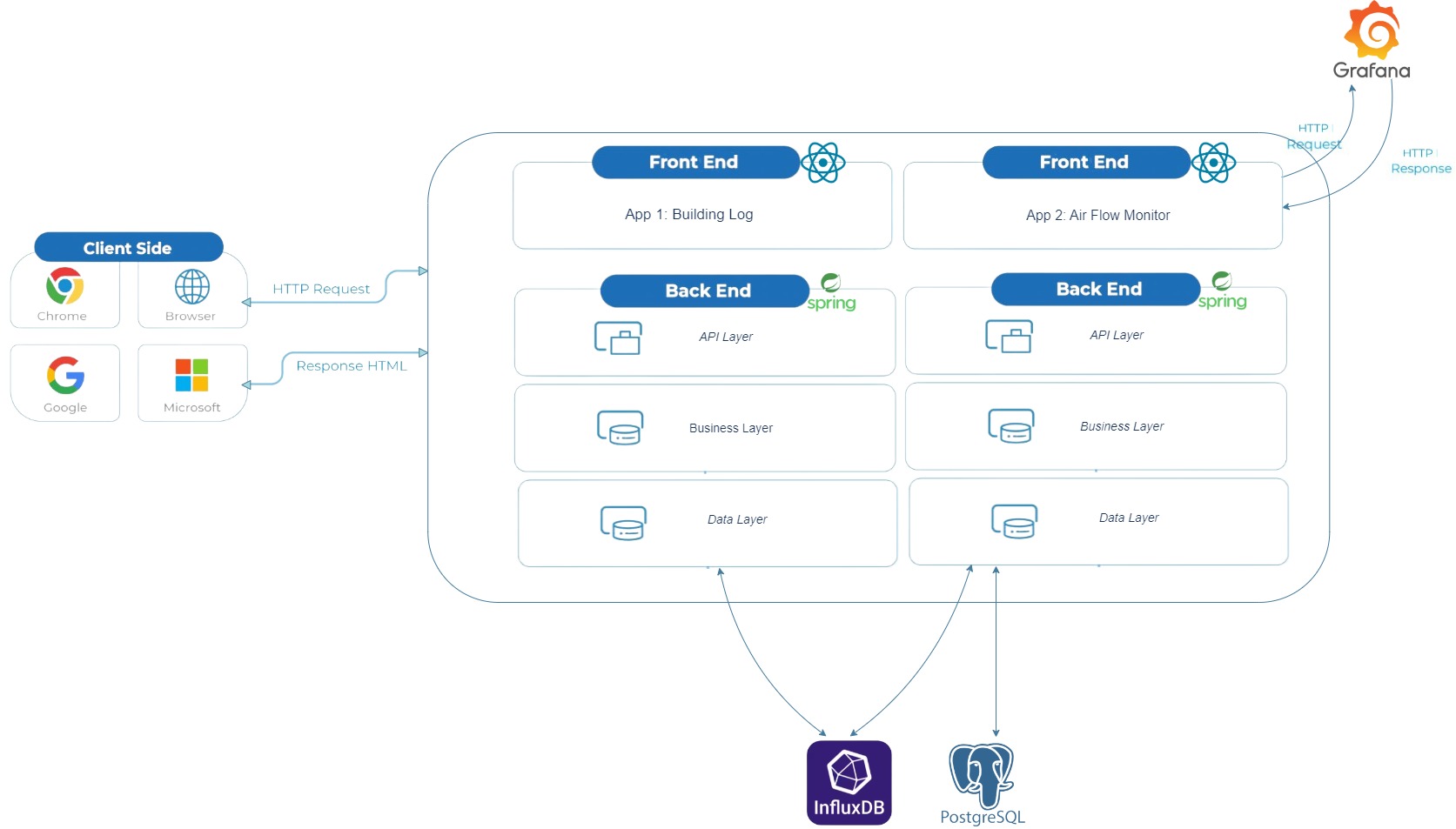
То нас доводи до следећег битног аспекта у овој бази података а то је анализа података.

Базе података временских серија представљају изузетно ефикасан инструмент за анализу података, посебно у контексту великог обима акумулираних информација. С обзиром на њихову могућност временског обележавања за сваки појединачни податак, ове базе обезбеђују дубок увид у динамику појединачних компоненти током времена. Анализом међусобног утицаја компоненти и откривањем шаблона током продужених временских периода, стварају се корисне информације и закључци.

Избор временске серијске базе података (ТСДБ) у контексту чувања догађаја уласка/изласка радника из пословне зграде произлази из низа могућности детаљне анализе података. Ова одлука обезбеђује темељ за складиштење свих релевантних података током времена, стварајући могућност за будућу надоградњу система кроз имплементацију специфичних упита који ће анализирати податке о уласцима и изласцима радника. Тренутно, ови упити нису имплементирани, али постоји потенцијал за њихово касније увођење.

# Архитектура система

Квалитетно осмишљена архитектура омогућава успостављање хармоничне интеграције различитих компоненти, обезбеђујући стабилност, скалабилност и оптималну оперативну ефикасност. Ово поглавље ће пажљиво анализирати кључне карактеристике архитектуре система имплементираног у оквиру ове студије.



Слика 2. Дијаграм архитектуре система

На дијаграму архитектуре система (слика 1) приказане су кључне компоненте које чине *BuildingLog* апликацију. Корисник има приступ *BuildingLog* апликацији која је специфично дизајнирана за симулацију улазака и излазака људи из пословне зграде. Клиентски дио апликације пружа корисницима једноставан приступ свим функционалностима система. Када корисник интерагује са сучељем, на пример, кликом на одређено дугме, клиентски дио апликације шаље *REST* захтјев ка серверском дијелу апликације.

Серверски дио апликације *BuildingLog* је у комуникацији са *InfluxDB* базом података. Приликом обраде захтева извршавају се одговарајући упити над *InfluxDB* базом података како би се добиле неопходне информације.

Све наведене компоненте система су прецизно повезане, чиме се постиже координисано функционисање *BuildingLog* апликације.

У оквиру имплементираног система, *AirFlowMonitor* апликација игра кључну улогу, обезбеђујући аутентификованим корисницима могућност надгледања очитаних вриједности са сензора ваздуха у пословној згради, као и обавјештавање радника о неочекиваним очитавањима. Слично *BuildingLog* апликацији, *AirFlowMonitor* такође поседује засебан клиентски и серверски дио.

Клиентски дио апликације *AirFlowMonitor* визуализује податаке о очитаним вриједностима са сензора, приказује актуелна обавјештења и омогућава кориснику приступ функционалностима система у зависноти од улоге корисника у систему. Комуникација између клиентског и серверског дијела апликације *AirFlowMonitor* одвија се путем *REST* захтева. Клиентски дио апликација шаље одређене параметре серверском дијелу, који затим извршава одговарајуће упите над *InfluxDB* или *PostgreSQL* базом података, како би добио релевантне податке.

У контексту визуализације података, клиентски дио *AirFlowMonitor*-а остварује везу са *Grafana* алатом, који омогућава генерисање графикона и визуализацију података. Клиентски дио шаље упит *Grafani*, прослеђујући јој параметре, а *Grafana* на основу прослијеђених вриједности генерише одговарајући графикон који се затим приказује на корисничком сучељу.

Важно је напоменути да комуникација између компоненти осим неком акцијом корисника, може да буде иницирана и појавом одређених вриједности у бази података. *InfluxDB* периодично извршава скрипту која провјерава да ли су све вриједности очитавања очекиване, уколико се појави нека неочекивана вриједности погоди се *REST* *endpoint* у контролеру серверског дијела апликације.

Затим се путем *WebSocket-а* који је успоставњен између серверског и клиентског дијела просљеђује обавјештење на корисничко сучеље. Отворен *WebSocket* канал омогућује серверском дијелу да шаље обавештења у реалном времену на основу догађаја који се дешавају у бази података.

У процесу аутентификације и ауторизације, коришћене су библиотеке као што су *json web token* за рад са *JSON* *Web* *Token*-има, док је за слање обавештења путем електронске поште коришћена библиотека *spring boot starter mail*.

# Пословна логика

У наставку, детаљно ће бити разматран кључни аспект имплементираног система – аларм систем. Предмет анализе биће начин функционисања овог аларм система, као и детаљи његове имплементације. Биће објашњено како систем обавештава раднике, које су специфичности његове реакције на различите догађаје и како су унапређене перформансе кроз имплементацију.

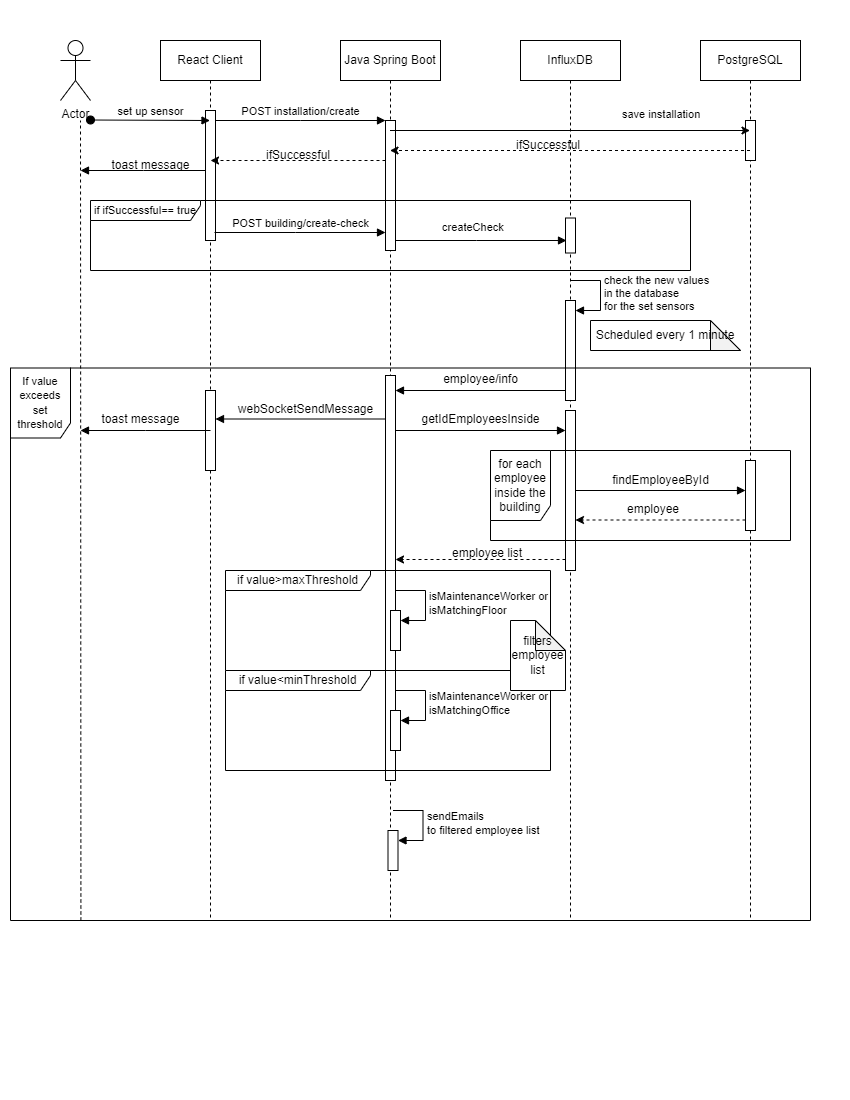
Овај аспект имплементације игра битну улогу у усмеравању радника одржавања система ка правовременом и одговарајућем реаговању на различите податке и догађаје. Анализом функционалности и детаља овог аларм система, ово поглавље ће пружити дубок увид у његову суштину и допринос у оптимизацији радних процеса у оквиру истраживаног информационог система.

## Имплементација аларм система

Динамички дијаграм секвенци, представљен на слици 1 састоји се из два дела.

У првом делу илустрован је процес креирања новог објекта *Installation*. Објекат *Installation* садржи информације о инсталацији сензора, конкретно који сензор је постављен, у којој канцеларији и када.

Овај дио система је важан за аларм систем из разлога што се при креирању нове инсталације креира *check* у *InfluxDB* бази података, а чекови представљају основу аларм система. У наставку ће бити разложене све битне компоненте за функционисање аларм система, представљене у другом дијелу дијаграма.



### *Check-ови у InfluxDB бази података*

*Check*-ови у *InfluxDB* бази података представљају кључни механизам за надзор и евалуацију стања података у временско-серијској бази података. Ови *Check*-ови се користе како би се дефинисали услови или правила која се примјењују на податке, што омогућава корисницима да аутоматски детектују и реагују на нежељене или необичне догађаје у временским серијама. То укључује прекорачења прагова вриједности или одсуство очекиваних података.

За исправну конфигурацију ове функционалности, неопходно је подесити три компоненте: чекови (*checks*), крајње тачке за обавјештавање (*notification endpoint*s) и правила за обавјештавање (n*otification rules*).

#### Чекови (checks)

Чекови представљају *Flux* скрипте које се извршавају на задатим временским интервалима. У овом систему, имплементацијом је постављен временски интервал на 1 минут, након којег се *Flux* скрипта изнова покреће. При дефинисању чека у скрипти, потребно је задати јединствени идентификатор чека, назив и тип. Затим, треба поставити прагове (*thresholds*) у односу на које систем упоређује измјерене вриједности. И наравно, потребно је дефинисати опсег података над којима ће се скрипта извршавати.

#### Прагови (*Thresholds*)

Прагови (*Thresholds*) представљају вредности које се постављају као границе или услови. Ови прагови се користе за дефинисање специфичних граница или опсега вриједности унутар временских серија података. Када податак премаши или падне испод постављених прагова, *Check* детектује одступања и може аутоматски покренути одговарајуће акције, као што су обавјештења или упозорења.

Такође, могуће је поставити различите услове у односу на вриједност прагова. Ови услови могу бити означени различитим статусима као што су *critical, warn, info,* или *ok*. Оваква конфигурација дозвољава прецизно искаљивање на који начин систем треба реаговати на различите нивое одступања од дефинисаних прагова.

Систем имплементира динамичко дефинисање вриједности прагова и креирање чекова, обезбеђујући флексибилност и адаптивност система према конкретним потребама и стандардима. У релационој бази података, за сваку инстанцу објекта сензора, чува се информација о минималним и максималним дозвољеним вриједностима очитавања за тај сензор. На пример, истраживања о температурама у канцеларијама показују оптимални опсег између 68 и 76 фаренхајта.

Када администратор система региструје нови сензор температуре у систему, има могућност задавања минималних и максималних вриједности очитавања. Ове вриједности аутоматски постају прагови у систему и користе се за креирање чекова. Администратор система такође има опцију накнадног мијења ових вриједности, а промјене ће бити ажуриране у вриједностима прагова у InfluxDB бази података. Ова функционалност пружа администратору система могућност да динамички адаптира конфигурацију система у складу са промјенама и изазовима у околини.

За креирање чекова, апликација користи библиотеку Influx Client. Ова библиотека омогућава конекцију на InfluxDB базу и пружа богат скуп функционалности за ефикасно управљање и праћење података.

Креирање чекова се изводи путем класа и функција из ове библиотеке. Коришћењем класа као што су ThresholdCheck, DashboardQuery, Threshold, GreaterThreshold, LesserThreshold, као и функција као што су setLevel, setValue, setQuery, setOrgID, setThresholds, апликација је у стању динамички дефинисати услове и параметре чекова.

#### Крајње тачке за обавјештавање (notification endpoints)

У контексту Checks-ова InfluxDB базе података, Notification endpoints представљају целе одредишне локације или сервисе на које се обавештења достављају у случају детекције одступања у подацима од стране Check-а. Ови endpoint-и могу укључивати различите комуникационе канале као што су Slack, HTTP endpoint-и и Pagerduty. У ситуацији када је изабрана HTTP опција, налази се неопходно дефинисати HTTP методу, методу аутентификације и URL.

С обзиром на постојање ауторизације и аутентификације у имплементираном систему, неопходно је било ефикасно решити изазов методе ауторизације захтева који пристижу од базе ка дефинисаном ендпоинту. Ова ситуација је решена путем избора опције "none" за методу аутентификације при креирању крајње тачке за обавештења, а на серверском дијелу апликације је на бијелу листу постављена коришћена путања */employee/info*.

#### Правила за обавјештавање (notification rules)

Notification Rule у InfluxDB представља функционалност која омогућава корисницима аутоматско покретање обавештења или акција када се испуне одређени услови у временским серијама података. Ова функционалност представља ефикасан начин за праћење и реаговање на значајне догађаје у реалном времену.

При дефинисању Notification Rule-а, корисници одређују "Check" као основу за обавештење. Check садржи услове које подаци морају задовољити како би покренули обавештење. Додатно, корисници конфигуришу параметре као што су "Every" за одређивање учесталости провере услова и "For" за одређивање колико узастопних периода провјере услова мора бити задовољено прије него што се покрене обавјештење. Коначно, корисници одређују "Endpoint" да би одредили где ће обавештење бити послато. Путем ових правила, корисници могу прилагодити параметре обавештења у складу са својим потребама, што значајно повећава аутоматизовано реаговање система на критичне или специфичне догађаје у временским серијама података.

### Дио пословне логике аларм система на серверској страни

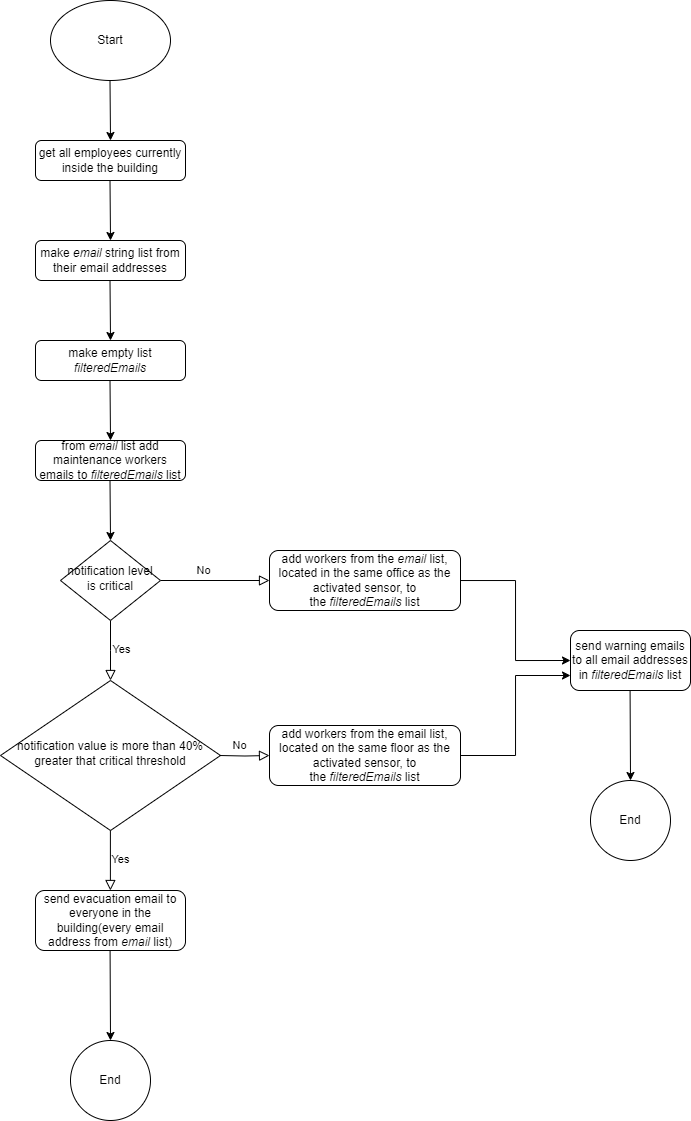
Када обавештење буде активирано, рест метода на путањи */employee/inf*o прима обавештење. Наставак описује последње кораке обраде обавештења и његово дистрибуирање крајњим корисницима.

Први корак у обради обавјештења на серверској страни је парсирање пристиглог обавјештења у објекат *Notification*.

Затим се парсирано обавјештење у реланом времену просљеђује клијентској страни апликације помоћу *web socket-а*.

С обзиром да систем подржава различите улоге у систему, претоставка је да корисник типа радник одржавања система треба да види сва обавјештења која дођу из базе података, те ће се кориснику који има ову улогу сва обавјештења приказати у виду *toast* порука. Радници осталих улога у систему биће обавјештени о само њима релевантиним обавјештењима путем електронске поште по алгоритму који ће бити објашњен у наставку.

Други корак је слање обавјештења радницима на емаил према алгоритму објашњеном дијаграмом активности.



Основна идеја алгоритма јесте да одреди за кога је важно које обавјештење и у којој мјери. Па тако, уколико се деси очитавање на сенору које је мање од доње границе оптималности о томе ће се поред радника одржавања система, oбавијесити и радници у тој канцеларији у којој се десило очитавање. Овакав тип обавјештења који стигне када се појави мања вриједност од минималног прага класификују се као упозорење (*warn*).

Ако се појави очитавање које је веће од горње границе оптималности одговарајућег сензора систем разликује два сценарија.

Први је ако очитане вриједности у мањој мјери прелазе максималну границу оптималности, тачније ако је прекорачење мање од 40% од максималну границу оптималности. Нпр, узмимо да опсег оптималности температуре између 68 и 76 фаренхајта, а у канцеларији на првом спрату TLM0100 се појави очитавање 85 фаренфајта. Ова вриједност од 85 фаренхајта је за 8,44% већа од максималне границе оптималности. О оваквој аномалити биће обавјештени сви радници који су тренунто у згради а раде на 1. спрату. Узима се да ова ситуација представља потенцијалну опасност и у складу са тим се упозоравају радници који раде у близини. Међутим, ако се деси прекорачење од 40% или веће, на примјер 106.4 фаренхајта, ова ситуација ће се тумачити као засигурно опасна и свим запосленим који су тренутно у згради ће се послати наређење о евакуацији.

Сваки део обавештавања и акција при препознавању различитих аномалија доприноси ефикасном и детаљном управљању ситуацијама у реалном времену.

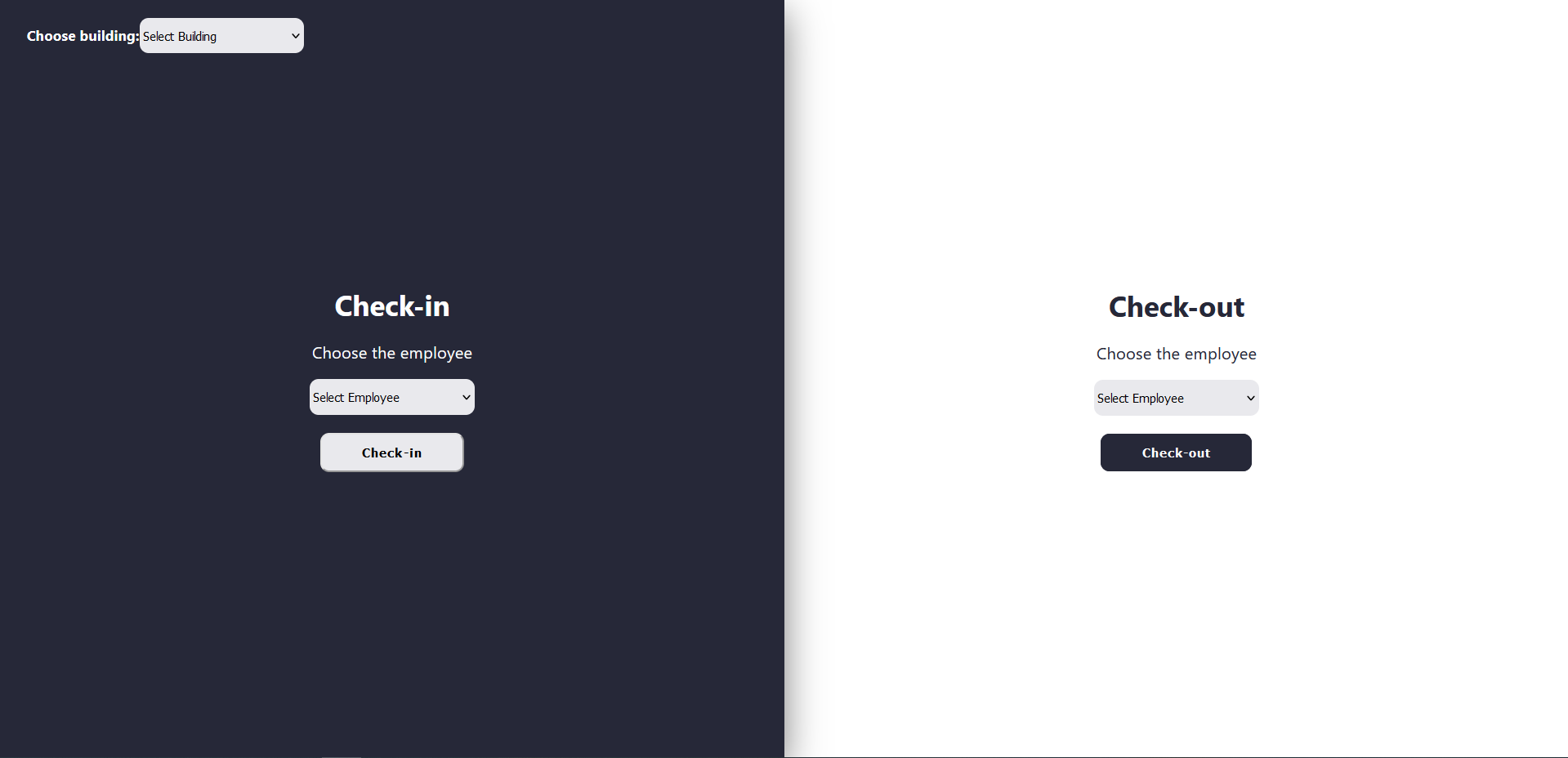
# Опис апликативног рјешења

У овом поглављу биће представљен кориснички интерфејс са освртом на појединачне целине и описом појединачних функционаности које су имплементиране у апликативном решењу.

## Апликација за симулирање улазка/излазка из пословне зграде

У овом поглављу биће представљена функционалност коју подржава апликација за симулирање уласка/изласка из пословне зграде, названа *BuilidngLog* апликација.

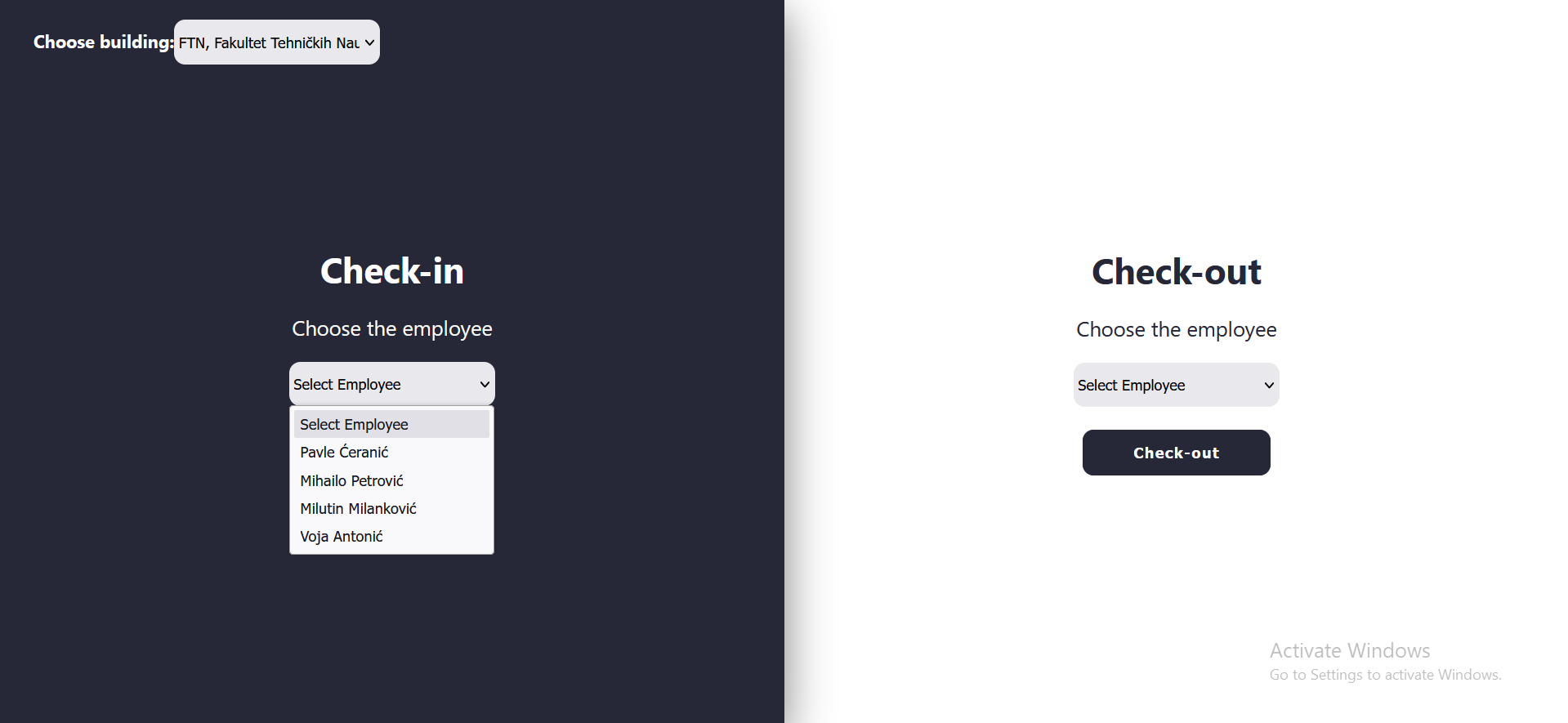
Апликација *BuildingLog* обухвата једну страницу, а дизајн ове странице приказан је на слици 1.



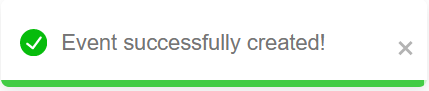
Слика 1.

Први корак у симулацији је одабир пословне зграде из *drop down meni*-a у горњем лијевом углу.

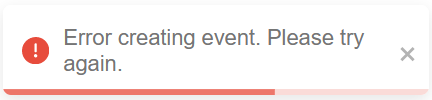
У следећем кораку, корисник би требало да изабере радника из *drop-down* менија, који се налази на лијевој или десној половини екрана, користећи лијеву страну ако жели симулирати улазак, а десну ако жели симулирати излазак.



После избора радника, трећи корак укључује клик на дугме *Check In* или *Check Out*.

Корисник ће бити информисан о успјешности симулације путем нотификације. У случају успешне акције, кориснику ће бити приказана порука о успјеху.

У случају неуспјешне акције, кориснику ће бити приказана порука о грешци.



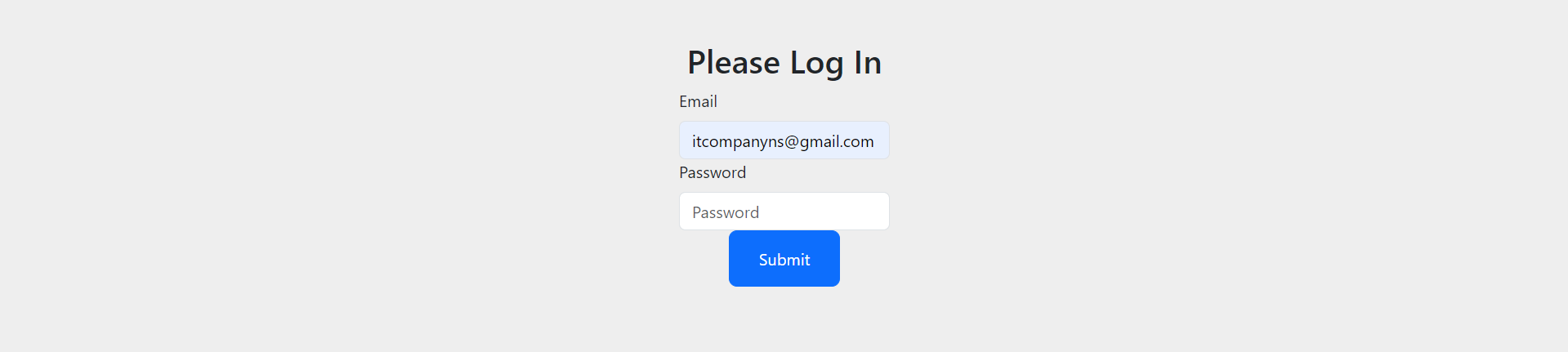
## Апликација за надгледање канцеларијског простора

У овом поглављу биће представљене функционалности које подржава апликација за надгледање канцеларијског простора, названа *AirFlowMonitor* апликација.

### Пријава на систем

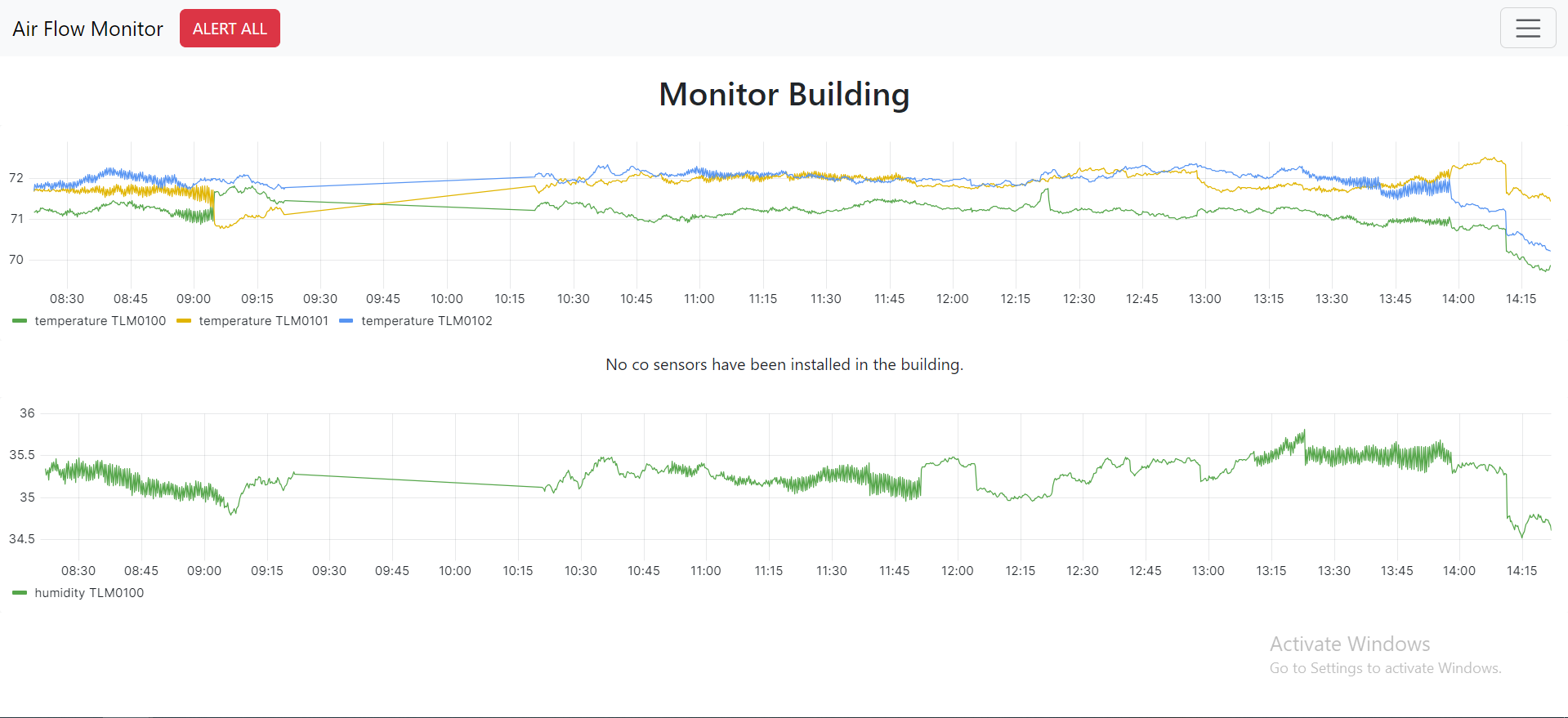
Да би корисник приступио систему мора бити претходно регистрован у систем од стране администратора. Регистрован корисник приступа систему уношењем своје *email* адресе и лозинке.

Форма ћи кориснику сугерисати да *email* адреса мора имати одређени формат и да оба поља морају бити попуњена да би акција била извршена.



### Надгледање параметара на нивоу зграде

Након успешне пријаве на систем, кориснику ће бити преусмерен на *Monitor Building* страницу. На овој страници, доступни су различити графикони који приказују вриједности очитавања са одређених типова сензора на нивоу цијеле зграде. На сваком графикону, подаци су представљени за све постављене сензоре одређеног типа, на пример, температуре. У легенди графикона, имена канцеларија у којима су сензори постављени, омогућавају кориснику лакше идентификовање и анализу вриједности.

Изглед *Monitor Building* странице за кориснике типа "радник одржавања система" и "админ" приказан је на слици 1. Док корисници типа "админ" и "радник одржавања система" имају видљиво дугме *Alert All*, корисници типа "радник" неће видети ово дугме на страници. Због ове мале разлике, слика изгледа странице за кориснике типа "радник" неће бити приложена. 

### Надгледање параметара на нивоу канцеларије

# Закључак

# Списак кориштених скраћеница

*REST*

*API*

# Литература

1. Why Time Series Matters For Metrics, Real-Time Analytics And, Sensor Data, Paul Dix, CTO and Founder, InfluxData
2. https://docs.influxdata.com/
3. ……….
4. *………*

# Биографија

Тамара Антић рођена је 23. маја 2000. године у Бањој Луци. Завршила је Гимназију у Баљој Луци, општи смјер. Школске 2019/2020. године уписао је Факултет Техничких Наука у Новом Саду, одсек Рачунарство и аутоматика, усмерење Примењене рачунарске науке и информатика. Положила је све испите предвиђене планом и програмом и стекла услов за одбрану завршног рада.