

**UNIVERSITETI I PRISHTINËS “HASAN PRISHTINA”**

**FAKULTETI I INXHINIERISË ELEKTRIKE DHE KOMPJUTERIKE**



# **Smart Health Monitoring System**

**Internet of Things - Master's Program**

***Professor:***

**Prof. Dr. Besmir Sejdiu**

***Students:***

**Lirim Maloku**

**Albion Ademi**

## Hyrje

Në këtë kapitull, do të prezantohet një pamje e përgjithshme e sistemit të monitorimit të shëndetit të zgjuar. Ky sistem është zhvilluar për të monitoruar dhe analizuar në kohë reale të dhëna shëndetësore nga sensorë të ndryshëm, për të ndihmuar në diagnozën dhe ofrimin e sugjerimeve për përmirësimin e shëndetit të pacientëve.

Sistemi përbëhet nga:

- Një shërbim dhe API të krijuara me .NET Core që simulon të dhëna dhe i dërgon ato në një topic të Kafka-s.
- Një pjesë përpunimi të të dhënave të realizuar me Spark Streaming duke përdorur PySpark, që dëgjon dhe përpunon të dhënat nga Kafka.
- Një bazë të dhënash Cassandra ku ruhen të dhënat e përpunuara.
- Një aplikacion në Angular që vizualizon të dhënat e ruajtura përmes grafikëve dhe përdor inteligjencën artificiale për të gjeneruar sugjerime.
- Poashtu përdoret një model i AI "llama2" për sugjerimet e gjeneruara.

Kodi burimor dhe të dhënat tjera rreth projektit mund të gjenden në këtë [lidhje](#).

## Dizajni i sistemit

Arkitektura e përgjithshme e sistemit dhe komponentët kryesorë:

- **Arkitektura e sistemit:** Përbëhet nga disa module që punojnë në harmoni për të mbledhur, përpunuar dhe vizualizuar të dhënat. Diagramet mund të përdoren për të ilustruar lidhjet mes komponentëve të ndryshëm.
- **Komponentët kryesorë:**
  - **API dhe shërbimi në .NET Core:** Përgjegjës për simulimin dhe dërgimin e të dhënave në Kafka.
  - **Kafka:** Shërben si një mesazheri për transmetimin e të dhënave midis shërbimit dhe Spark Streaming.
  - **Spark Streaming me PySpark:** Përpunon të dhënat në kohë reale dhe i ruan në bazën e të dhënave Cassandra.
  - **Baza e të dhënave Cassandra:** Magazina për ruajtjen e të dhënave të përpunuara.
  - **Aplikacioni Angular:** Shfaq të dhënat në grafikë dhe përdor AI për sugjerime.

## Veglat dhe teknologjitë e përdorura

Veglat dhe teknologjitë e përdorura në projekt:

- **.NET Core:** Për zhvillimin e shërbimeve API për simulimin e të dhënave dhe leximin e tyre për vizualizim.
- **Kafka:** Një platformë e fuqishme për transmetimin e të dhënave në kohë reale.
- **PySpark dhe Spark Streaming:** Për përpunimin dhe analizën e të dhënave në kohë reale.
- **Cassandra:** Baza e të dhënave NoSQL për ruajtjen e të dhënave me shkallëzim të lartë.


















- **Docker:** Përdoret për orkestrimin dhe menaxhimin e mjedisit të gjithë komponentëve të sistemit.
- **Angular:** Përdoret për zhvillimin e aplikacionit front-end për vizualizimin e të dhënave.
- **Inteligjenca Artificiale:** Përdoret për gjenerimin e sugjerimeve bazuar në të dhënat e mbledhura.
- **Kafdrop:** Përdoret për vizualizim të topics të Kafka.
- **TablePlus:** Përdoret për menaxhim të Cassandra db.
- **DockerDesktop:** Aplikacion që lejon zhvilluesit të krijojnë, testojnë dhe menaxhojnë kontejnerë në një mjedis lokal me lehtësi.
- **OLLama:** Modeli për inteligjencën artificiale ([LINK](#)).

## Demonstrimi dhe zbatimi i sistemit

Ky kapitull do të përshkruajë procesin e zbatimit hap pas hapi të sistemit. Pas shkarkimit të projektit nga linku në GitHub hapim terminalin në lokacionin ku ndodhen të dhënat e projektit. Fillimisht ekzekutojmë komandat e nevojshme për Docker:

**docker-compose build --no-cache** dhe pastaj **docker-compose up**

Kjo krijon docker images dhe container të cilët vënë në punë veglat në figurën e mëposhtme.

 <u>smarthealthmonitorin</u>		
	<u><a href="#">spark-master-1</a></u> 8a4f5457c381 	<u><a href="#">bitnami/spark:3.3.1</a></u>
	<u><a href="#">zookeeper-1</a></u> d093bdcdeef 	<u><a href="#">zookeeper:3.6.3</a></u>
	<u><a href="#">cassandra-1</a></u> 3f4a1fa058ef 	<u><a href="#">cassandra:3.11</a></u>
	<u><a href="#">spark-worker-1</a></u> 72ba05ff0581 	<u><a href="#">bitnami/spark:3.3.1</a></u>
	<u><a href="#">cassandra-init-1</a></u> ea49d051d9e2 	<u><a href="#">cassandra:3.11</a></u>
	<u><a href="#">kafka-1</a></u> 33c00543262b 	<u><a href="#">confluentinc/cp-kafka:7.0.1</a></u>
	<u><a href="#">spark-submit-1</a></u> 88d6d1cf298d 	<u><a href="#">smarthealthmonitoring-spark-submit</a></u>
	<u><a href="#">kafdrop-1</a></u> 7002955aca14 	<u><a href="#">obsidiandynamics/kafdrop</a></u>

Përveç këtyre veglave poashtu ekzekutojmë një skriptë të Cassandra-s që i krijon tabelat e nevojshme, në keyspace të quajtur “health\_data” dhe bën insertimin e disa “master data” në disa prej tabelave. P.sh insertojmë 5 sensorë në tabelën sensor.

Items	Queries	History	sensor_code	manufacturer	sensor_name
Search for item...			SENS005	SmartHealth Systems	Advanced Health Monitor
			SENS003	BioHealth Devices	Environmental Sensor
alarm			SENS004	WellBeing Technologies	Multi-Parameter Sensor
patient			SENS002	HealthCorp Ltd.	Health Monitoring Device
sensor			SENS001	MedTech Inc.	Vital Sign Monitor
sensor_data					
sensor_node					

Ndërsa mëposhtë mund të shohim tabelat e përdorura dhe skemën.

**Table: sensor (sensor\_code TEXT PRIMARY KEY, sensor\_name TEXT, manufacturer TEXT)**

**Table: patient (patient\_id UUID PRIMARY KEY, first\_name TEXT, last\_name TEXT, birthday DATE, gender TEXT, address TEXT)**

**Table: sensor\_node (node\_id UUID PRIMARY KEY, node\_name TEXT, battery\_percentage INT, hospital\_name TEXT, patient\_id UUID, sensor\_code TEXT)**

**Table: sensor\_data (sensor\_node\_id UUID, time\_stamp TIMESTAMP, pulse\_rate INT, body\_temperature FLOAT, room\_temperature FLOAT, room\_humidity FLOAT, PRIMARY KEY (sensor\_node\_id, time\_stamp))**

**Table: alarm (alarm\_id UUID PRIMARY KEY, time\_stamp TIMESTAMP, alarm\_cause TEXT, alarm\_cause\_value TEXT, alarm\_description TEXT, sensor\_node\_id UUID)**

Pasi të përfundojë ekzekutimi i komandave në Docker, skripta e Python e cila përdor PySpark fillon dëgjimin në Kafka topic me emrin ‘sensor-topic’.

```
2024-09-22 20:32:54 :: loading settings :: url = jar:file:/opt/bitnami/spark/jars/ivy-2.5.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
2024-09-22 20:32:57 INFO: __main__: Starting spark session...
2024-09-22 20:33:00 INFO: __main__: Spark Session started.
2024-09-22 20:33:00 INFO: __main__: Reading from Kafka topic...
2024-09-22 20:33:03 INFO: __main__: Reading messages from Kafka topic kafka:9992
2024-09-22 20:33:04 INFO: __main__: Dataframe[sensor_node_id: string, time_stamp: string, pulse_rate: int, body_temperature: float, room_temperature: float, room_humidity: float]
2024-09-22 20:33:04 INFO: py4j.java_gateway: Callback Server Starting
2024-09-22 20:33:04 INFO: py4j.java_gateway: Socket listening on ('127.0.0.1', 45679)
```

Pasi që skripta po dëgjon në topic të Kafka-s, tani mund të përdorim API e krijuar në .NET Core për gjenerimin e të dhënave ashtu që të tentojmë të simulojmë një sensor real. Për këtë përdorim pikë fundore si më poshtë dhe si parametër kemi numrin e rreshtave që duam të gjenerojmë.

## Simulator


**POST** /api/Simulator/GenerateData

Parameters

Try it out

Name	Description
nrOfRows	nrOfRows
integer(\$int32)	
(query)	

Poashtu përdorim veglën Kafdrop që shërben për vizualizimin më të lehtë të mesazheve në Kafka.

Star

## Topic Messages: sensor-topic

First Offset: 0 Last Offset: 1500 Size: 1500

Partition: 0 Offset: 0 # messages: 100 Key format: DEFAULT Message format: DEFAULT

View Messages

⏮ ⏪ ⏩ ⏭

Offset: 0 Key: empty Timestamp: 2024-09-22 17:40:50.392 Headers: empty

⚡ {"sensor\_node\_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef", "time\_stamp": "2024-09-22 17:40:50", "pulse\_rate": 73, "body\_temperature": 35.4, "room\_temperatur

Offset: 1 Key: empty Timestamp: 2024-09-22 17:40:50.394 Headers: empty

⚡ {"sensor\_node\_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef", "time\_stamp": "2024-09-22 17:41:50", "pulse\_rate": 47, "body\_temperature": 33.8, "room\_temperatur

Offset: 2 Key: empty Timestamp: 2024-09-22 17:40:50.394 Headers: empty

⚡ {"sensor\_node\_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef", "time\_stamp": "2024-09-22 17:42:50", "pulse\_rate": 30, "body\_temperature": 44.7, "room\_temperatur

Offset: 3 Key: empty Timestamp: 2024-09-22 17:40:50.394 Headers: empty

⚡ {"sensor\_node\_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef", "time\_stamp": "2024-09-22 17:43:50", "pulse\_rate": 97, "body\_temperature": 38.1, "room\_temperatur

Offset: 4 Key: empty Timestamp: 2024-09-22 17:40:50.394 Headers: empty

⚡ {"sensor\_node\_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef", "time\_stamp": "2024-09-22 17:44:50", "pulse\_rate": 77, "body\_temperature": 36.4, "room\_temperatur

Offset: 5 Key: empty Timestamp: 2024-09-22 17:40:50.394 Headers: empty

⚡ {"sensor\_node\_id": "a1b2c3d4-e5f6-7890-1234-567890abcdef", "time\_stamp": "2024-09-22 17:45:50", "pulse\_rate": 75, "body\_temperature": 40.1, "room\_temperatur

Prishtinë, 2024

Të dhënat e gjeneruara në kohë reale procesohen dhe ruhen në Cassandra.

```
2024-09-22 19:40:57 INFO:py4j.clientserver:Received command c on object id p0
2024-09-22 19:40:57 INFO:__main__:Processing batch 2
2024-09-22 19:40:57 INFO:__main__:Number of rows before writing to Cassandra: 1425
2024-09-22 19:40:57 INFO:__main__:Sensor data for batch 2 inserted into sensor_data table successfully.
2024-09-22 19:40:57 batch df show inside alarms
2024-09-22 19:40:57 +-----+-----+-----+-----+-----+-----+
2024-09-22 19:40:57 | sensor_node_id| time_stamp|pulse_rate|body_temperature|room_temperature|room_humidity|
2024-09-22 19:40:57 +-----+-----+-----+-----+-----+-----+
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 18:55:50|67|35.0|22.7|53.8|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 18:56:50|92|31.3|30.3|47.0|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 18:57:50|123|44.0|33.8|24.9|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 18:58:50|54|35.7|12.2|26.1|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 18:59:50|130|38.5|40.5|77.5|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:00:50|56|43.9|36.4|65.2|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:01:50|87|35.5|43.2|30.0|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:02:50|125|35.9|20.6|59.9|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:03:50|122|37.3|34.5|41.2|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:04:50|42|44.2|33.4|42.4|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:05:50|110|42.9|27.7|35.2|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:06:50|111|44.1|15.7|48.9|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:07:50|136|42.5|34.0|28.8|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:08:50|76|32.9|31.7|31.9|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:09:50|140|30.7|35.0|65.1|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:10:50|100|38.9|12.3|72.1|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:11:50|94|32.9|40.9|72.6|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:12:50|138|34.4|39.8|34.3|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:13:50|32|44.0|25.7|63.1|
2024-09-22 19:40:57 |a1b2c3d4-e5f6-789...|2024-09-22 19:14:50|122|40.3|22.5|67.3|
2024-09-22 19:40:57 +-----+-----+-----+-----+-----+-----+
2024-09-22 19:40:57 only showing top 20 rows
2024-09-22 19:40:57
2024-09-22 19:40:57
Items Queries History
Search for item...
alarm
patient
sensor
sensor_data
sensor_node
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:40:51.000+0000 32.5 90 60.3 22.4
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:41:51.000+0000 41.4 111 78.3 29
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:42:51.000+0000 31.4 68 73.5 40.6
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:43:51.000+0000 42.4 39 49.3 35.3
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:44:51.000+0000 40.7 86 64.2 15.6
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:45:51.000+0000 30.5 122 76.3 37.5
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:46:51.000+0000 44.4 121 20 30.3
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:47:51.000+0000 42.3 133 60.7 35
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:48:51.000+0000 36.6 111 68.2 34.7
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:49:51.000+0000 40.9 128 22.4 34.4
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:50:51.000+0000 32.3 81 66.4 37.3
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:51:51.000+0000 38.3 91 38.9 23.8
12345678-90ab-cdef-1234-567890abcdef 2024-09-22 17:52:51.000+0000 40.9 85 22.5 37.4
```

Zakonisht kur kemi të dhëna në rang jo normal duhet të aktivizohen disa alarme të cilat mund të jenë mesazhe, thirrje etj. Ne i kemi ruajtur alarmet në një tabelë të veçantë si më poshtë:

Items	Queries	History	alarm_id	alarm_cause	alarm_	alarm_description	sensor_node_id	time_stamp
			507d915a-9042-4d3b-9058-253e21c7d5a4	pulse_rate	112.0	Pulse rate is too high	d4e5f678-9012-3456-7890_	2024-09-22 18:30:51.000+0000
			08a5662f-ad5d-4388-b563-bb125adb36a8	pulse_rate	115.0	Pulse rate is too high	01234567-89ab-cdef-1234_	2024-09-22 19:58:51.000+0000
alarm			81b9b136-657d-4374-8781-310706a407ef	body_temperature	38.9	Temperature is too high	f6789012-3456-7890-abcd_	2024-09-22 19:23:51.000+0000
patient			ac9badb4-4dac-4669-a4d3-b3313ad454eb	room_humidity	79.4	Room humidity is too high	34567890-1bcd-ef23-4567_	2024-09-22 18:20:51.000+0000
sensor			570a7030-4914-403a-b8a0-d06b4f084b71	room_temperature	25.0	Room temperature is too high	d4e5f678-9012-3456-7890_	2024-09-22 18:46:40.000+0000
sensor_data			8e1438ca-ec35-441e-bf9f-1c6e08c154de	room_humidity	73.1	Room humidity is too high	a1b2c3d4-e5f6-7890-1234_	2024-09-22 19:50:50.000+0000
sensor_node			4350366a-be92-46e9-9c78-a5167b332cb3	room_temperature	43.1	Room temperature is too high	c3d4e5f6-7890-1234-5678_	2024-09-22 17:53:51.000+0000
			1ea287df-e40d-43b4-99fd-577404d15fa6	body_temperature	31.0	Temperature is too low	f6789012-3456-7890-abcd_	2024-09-22 19:45:51.000+0000
			b43a6fb8-d3f5-4120-8c39-8868e9b9ee99	room_temperature	27.1	Room temperature is too high	34567890-1bcd-ef23-4567_	2024-09-22 18:07:51.000+0000
			b74f1631-1e38-4b53-986a-f01cda2ac4d	pulse_rate	56.0	Pulse rate is too low	12345678-90ab-cdef-1234_	2024-09-22 17:56:51.000+0000
			959dbd29-ef35-41bb-8c64-55379bcc4a2a	room_temperature	19.1	Room temperature is too low	12345678-90ab-cdef-1234_	2024-09-22 18:17:51.000+0000
			9f057013-f0c4-4877-8577-ba5004d07017	room_temperature	12.3	Room temperature is too low	23456789-0abc-def1-2345_	2024-09-22 19:29:51.000+0000
			66540e43-8dfb-4910-9197-dfdc5824df90	room_temperature	34.7	Room temperature is too high	01234567-89ab-cdef-1234_	2024-09-22 18:46:40.000+0000
			adb9075d-6486-4e4b-9cba-b5257eab27dc	body_temperature	31.2	Temperature is too low	34567890-1bcd-ef23-4567_	2024-09-22 18:48:51.000+0000

Pra kemi të dhëna se prej cilit sensor është shkaktuar alarmi, në çfarë kohe, dhe poshtu se cili parametër dhe me çfare vlere e ka shkaktuar atë. Logjika për klasifikimin e alarmeve gjendet në skriptën në Python dhe duket si më poshtë. Pastaj normalisht bëhet insertimi i këtyre rreshtave në tabelën 'alarm' në Cassandra.

```
def generate_alarms(batch_df):
    alarms = []

    alarm_conditions = [
        ("body_temperature", 35.5, 37.0, "Temperature is too low", "Temperature is too high"),
        ("room_temperature", 20.0, 24.0, "Room temperature is too low", "Room temperature is too high"),
        ("room_humidity", 30.0, 60.0, "Room humidity is too low", "Room humidity is too high"),
        ("pulse_rate", 60, 100, "Pulse rate is too low", "Pulse rate is too high")
    ]

    for column, low_threshold, high_threshold, low_msg, high_msg in alarm_conditions:
        temp_alarms = batch_df \
            .withColumn("alarm_description",
                when(col(column) < low_threshold, low_msg)
                .when(col(column) > high_threshold, high_msg)) \
            .filter(col("alarm_description").isNotNull()) \
            .withColumn("alarm_id", generate_uuid()) \
            .withColumn("alarm_cause", lit(column)) \
            .withColumn("alarm_cause_value", col(column)) \
            .select("alarm_id", "time_stamp", "alarm_cause", "alarm_cause_value", "alarm_description", "sensor_node_id")

        logger.info(f"Printing alarms for condition {column[0]}:")
        temp_alarms.show()


        if temp_alarms.count() > 0:
            alarms.append(temp_alarms)
        else:
            logger.info(f"No alarms generated for {column}.")


    print(alarms)


    if alarms:
        combined_alarms = alarms[0]
        for alarm_df in alarms[1:]:
            combined_alarms = combined_alarms.union(alarm_df)
            print("combined alarms")
            combined_alarms.show(truncate=False)
        return combined_alarms
    return None
```


Pasi të kemi gjeneruar të dhëna, mund të përdorim ndërfaqen e krijuar në Angular për të analizuar të dhënat më lehtë. Kjo faqe përmban kartat me sensorët dhe të dhënat e tyre. Në figurën e mëposhtme


shohim një nga kartat.

 MedTech Inc. - Vital Sign Monitor

 Node A - SENS001 | 85%

 John Doe | 123 Health St, City A

 Node F - SENS001 | 90%

 Sophia Martinez | 303 Health Way,  
City F



Poashtu kemi të dhëna tjera rreth sensorëve mirëpo të vizualizuar përmes grafeve.

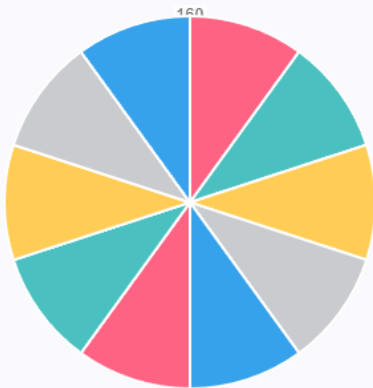
Number of nodes for each sensor

Vital Sign Monitor    Health Monitoring Device    Environmental Sensor    Multi-Parameter Sensor    Advanced Health Monitor



Number of measurements for each sensor node

Node A    Node B    Node C    Node D    Node E    Node F    Node G    Node H    Node I    Node J

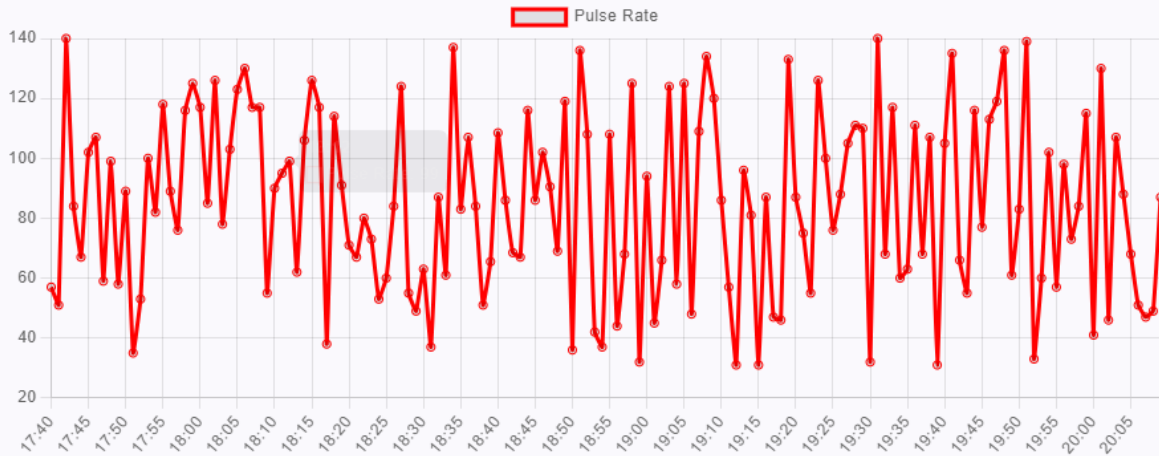


Pastaj vizualizimet tjera ndërlidhen me një nyje specifike sensorike.



Integrimi dhe përdorimi i inteligjencës artificiale është kryer duke përdorur modelin llama2, dhe pas gjenerimit të një pergjigjeje që arrihet nga shtypja e butonit në figurë, poshtë tij shfaqet pergjigja e gjeneruar nga modeli për gjenerim të tekstit.

### Pulse rate over time



#### Get AI Suggestion about pulse rates

A normal pulse rate for an adult is between 60-100 beats per minute (bpm). However, it's important to note that a normal pulse rate can vary depending on factors such as age, gender, physical activity level, and overall health.

Poashtu kemi dy CSV me të dhëna për tabelat alarm dhe sensor\_data, të cilat mund t'i përdorim për populim të tabelave, pra nëse duam të shohim pjesën e vizualizimeve pa e përdorur Spark Streaming dhe Kafka. Për këtë qëllim përdorim pikën fundore si në figurën e mëposhtme.

### IoT-Health-Monitoring 1.0 QAS3

/swagger/v1/swagger.json

#### Data

GET /api/Data

POST /api/Data/AddSensorAndAlarmData

#### Parameters

No parameters

#### Responses

Code	Description
200	Success

Links
No links

## Përfundime

Rezultatet e këtij projekti pra të dhënat dhe vizualizimet normalisht se mund të arrihen edhe me teknologji tjera e ndoshta më shpejtë e më thjeshtë. Mirëpo qëllimi ka qenë demonstrimi dhe përfitimi i përvojës me veglat dhe teknologjitë që përdorin sensorët, për të cilat kemi diskutuar përgjatë ligjëratave dhe ushtrimeve gjatë semestrit. Përgjatë semestrit dhe punës në projekt kemi kuptuar se përse

përdorim Kafkan që është sistem i fuqishëm i mesazheve që lejon transmetimin e të dhënave në kohë reale e që mund të përpunojë dhe ruaj një volum të madh mesazhesh me vonesa të ulëta, duke e bërë të përshtatshëm për sensorët që prodhojnë të dhëna vazhdimisht. Poashtu Cassandra që është një sistem të dhënash të shpërndara, i projektuar për të menaxhuar të dhëna të mëdha me performancë të lartë dhe disponueshmëri të lartë. Ndërmjet këtyre Spark Streaming ofron mundësinë për të përpunuar të dhëna në kohë reale, duke lejuar analizat e menjëhershme dhe reagimin ndaj ngjarjeve në kohë reale. Kjo është e rëndësishme për aplikacione që kërkojnë analizë të shpejtë dhe vendimmarrje të shpejtë. Integrimi me Kafka dhe Cassandra e bën të lehtë përpunimin e të dhënave që mbërrijnë nga sensorët dhe ruajtjen e rezultateve në bazën e të dhënave për analizë të mëtejshme. Poashtu kemi integruar inteligjencën artificiale e cila mund të përdoret në mënyra të ndryshme mirëpo ne e kemi shfrytëzuar për gjenerimin e sugjerimeve në të të dhënave të ruajtura të cilat po i përdorim për vizualizim.