Title: StoreMetrics

Participants: Kasper R, Kasper K & Jonas N.

Specs

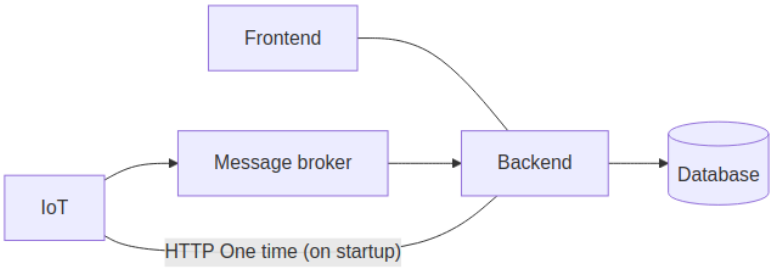
1. **Project Description:** We want to create a system to collect data from physical stores. The data we are interested in collecting are:

   - How many people have entered the store
   - How many products have been sold
   - What types of products have been sold
   - The total value of sold products
   - How many packages have been picked up and delivered to the store if the store has a package delivery spot

   The purpose is to collect data from stores and display it in an intuitive way where a user can filter and search the data.

2. **Domain Model:**

   - IoT devices: Collect data continuously and send it every minute.
   - Server: Runs an nginx web server that directs traffic to a backend written in PHP using the Laravel framework.
   - Backend: Receives data through a message broker and stores it in a MariaDB SQL server.
   - Frontend: Uses Laravel blade templates to display data.



3. **Functional Requirements:**

   - IoT devices must display an ID on a display, which can be used to filter the data.
   - IoT devices must only use HTTP for the initial communication to get a UUID.
   - IoT devices must be able to collect and send data continuously.
   - The server must be able to receive and store data from IoT devices.
   - The frontend must be able to display the collected data.
   - The frontend must have an admin page:
     - Login + logout.

    - Where you can group devices as a single unit.
    - Manage grouped devices.
  - Data display must include filtering and searching options.
  - All data must be accessible to everyone.
  - CI/CD Deployment.

4. **Non-functional Requirements:**
   - Usability: The user interface must be intuitive and easy to navigate.
   - Reliability: The system must have an uptime of 99.9%.
   - Performance: The system must handle up to 1000 concurrent devices.
   - Supportability: The code must be well-documented and easy to maintain.

5. **Limitations:**
   - Data predictions
   - Data pagination
   - Data export to csv

Tech stack

To collect the data from the stores, we plan to use IoT devices:

- The IoT devices will be collecting data contiuesly, and sent it to us every hour, however for this demo we're using Arduino Oplá, which isn't designed for our case, so we're randomly generating the data, to prove that the system can handle the data.
- The IoT devices will primarily transmit data over the mqtt protocol.
- The IoT devices will on initial setup use http to get some identification (uuid), so that the data sent from one device doesn't interferer with other devices.

To handle the data we plan to create a server:

- The server will be running nginx webserver, which will direct the trafic to our primary backend written in php using the Laravel framework.
- The backend will store the data received through our message broker, in a MariaDB sql server.
- To ensure that updating the servers doesn't break anything, we also want to setup extenxive testing of all endpoints to ensure that the code is working.

For the frontend, we intend to use blade.php:

- The site will have to primary views, the first is to get an overview of devices, and the second is to view the data collected.
- The data view will include options for filtering and searching the data.

IoT:

- Arduino
- c++

Message broker:

- RabbitMQ

Backend:

- php
- Laravel

Database:

- MariaDB

Frontend:

- Laravel blade templates

Database ER diagram

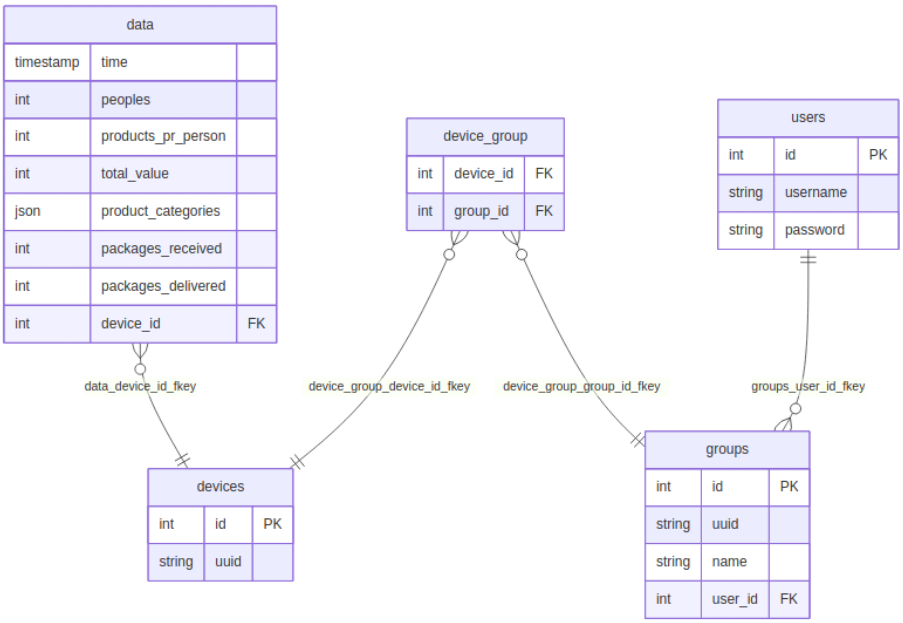


Figure 1: diagram

User guide

The guide is designed for https://krc-coding.dk

as a user you can:

Get all data:

just click the “Search” button with the text field empty. (might not currently work as there is to much data)

Get data from a specific device:

if you want to get data from a specific device (normaly your own device) you will have to type in the UUID of the device before clicking search.

(device UUID’s are displayed on the device screen)

here are some device UUID’s you can try:

- 63ea5a6f-506d-36e2-a711-3d96e1fc8b7b
- 46b2c477-91ec-34f4-aba0-acb4f1c4d48f

Sort the data:

after fetching some data just click on the colum header that you want the data sorted by, first click will sort in decenting order (biggest number first) and the second ascending. (smallest number first)

Signup/Login:

just click the “Login” button in the top right corner and login.

if you dont already have an account click the signup button and create one.

User profile:

once logged in the login button will be replaced by a button that will lead you to your profile.

Create a group:

on the profile page just type a name into the text field and click the “Create Group” button. after a group is created you now have a row that states the name of the group, what devices are in it, some actions you can take and the Group UUID

(this makes it so that you can get data from multiple devices at the same time without having to get all data)

Add device to group:

to add a device to a group you need to type in the UUID of the device into the text field in the “Action” collum, in the row of the group you want to add it to and click the “Add Device” button.

(you can see the UUID's of devices thats in the group undet the collum "Devices")

here are some device UUID's you can try:

- 63ea5a6f-506d-36e2-a711-3d96e1fc8b7b
- 46b2c477-91ec-34f4-aba0-acb4f1c4d48f

Remove device from group:

simply finde the device you wonna Remove on the Device list and click remove you will then be prompted it you are sure clicking ok will remove it.

(removed devices can be added back later so dont worry to much about accidently removing one)

Delete Group:

under the "Actions" collum click the "Delete Group" button you will then be prompted if you are sure.

(this will permanently delete the group so if you want the group back you will have to create a new one)

Get group data:

now if you go back to the search page (button in the top left corner) and put in your group uuid it before clicking search you will now get the data from all devices in the group.

(you dont need to be logged in to use the group UUID's)

if you deleted your group you can use mine:

- 68ed957b-c465-401e-9899-5977275bdcc0

Logout:

to log out go the your profile page and click the "Logout" button in the top right corner.

(this will automaticly send you back to the search page)

Dev setup guide

Requirements

- php 8.3.x NTS (Not Thread Safe)
- Composer
- MariaDB

Setup

1. **Git clone**
   - Clone the project down on your machine
2. **Install/update composer in project**
   - Using a terminal to navigate to the project and go into /StoreData-Collection
   - Run: `composer install`
3. **Env**
   - In terminal run both:
     - `cp .env.example .env`
     - `php artisan key:generate`
   - Update your DB settings to fit your database
4. **Database**
   - Create a new database with the same name as you used in your env
   - In the terminal run: `php artisan migrate`
5. **Run degub**
   - Run: `php artisan serve`

Debugging

Frontend

You can debug by going to localhost:8000

Backend

You need to send http request. We recommend using postman, as we have an collection in the files called: H5.postman_collection.json

Integration testing

Backend as an integration test setuped.

- **This is not necessary**. In testing we can use another db if we don't want to use our live:
  - In terminal run: `cp .env.testing.example .env.testing`
  - Update the db settings and copy APP_KEY from .env
  - Create the db on your db
  - Run: `php artisan migrate --env=testing`
- To try it run: `php artisan test`

Arduino setup guide (IoT device)

All Arduino related code can be found inside the Arduino folder.
All commands and info assumes that you've opened the Arduino folder.

Requirements:

Board type: `Arduino MKR WiFi 1010`

Libraries to install for arduino code to function:

- Arduino_MKRIoTCarrier by Riccardo Rizzo, Jose García, Pablo Marquínez
- WiFiNINA by Arduino
- NTPClient by Fabrice Weinberg
- ArduinoHttpClient by Arduino
- PubSubClient by Nick O'Leary
- ArduinoJson by Benoit Blanchon

Before upload:

1. Copy SECRETS.h.example to SECRETS.h: `cp SECRETS.h.example SECRETS.h`
2. Add your wifi info, if you're hosting the backend from a local pc, then the arduino should be on the same wifi.
3. Add the ip of the backend to the server host variable.
4. If the port of the backend is different from 8000, then update the server port variable as well.

After upload:

The arduino should now begin to connect to the required services.
If it fails to connect to the mqtt host, then it will display it on the primary display of the device.

To get the messages sent to the mqtt host into the database:

1. Open a new terminal on the pc/server running the backend.
2. Navigate in to the folder containing the backend code. `StoreDataCollection`
3. Run the following artisan command: `php artisan rabbitmq:consume`

This will start a process that will continue to listen for messages on the rabbitmq server.
To stop it from listening, simply stop the command.