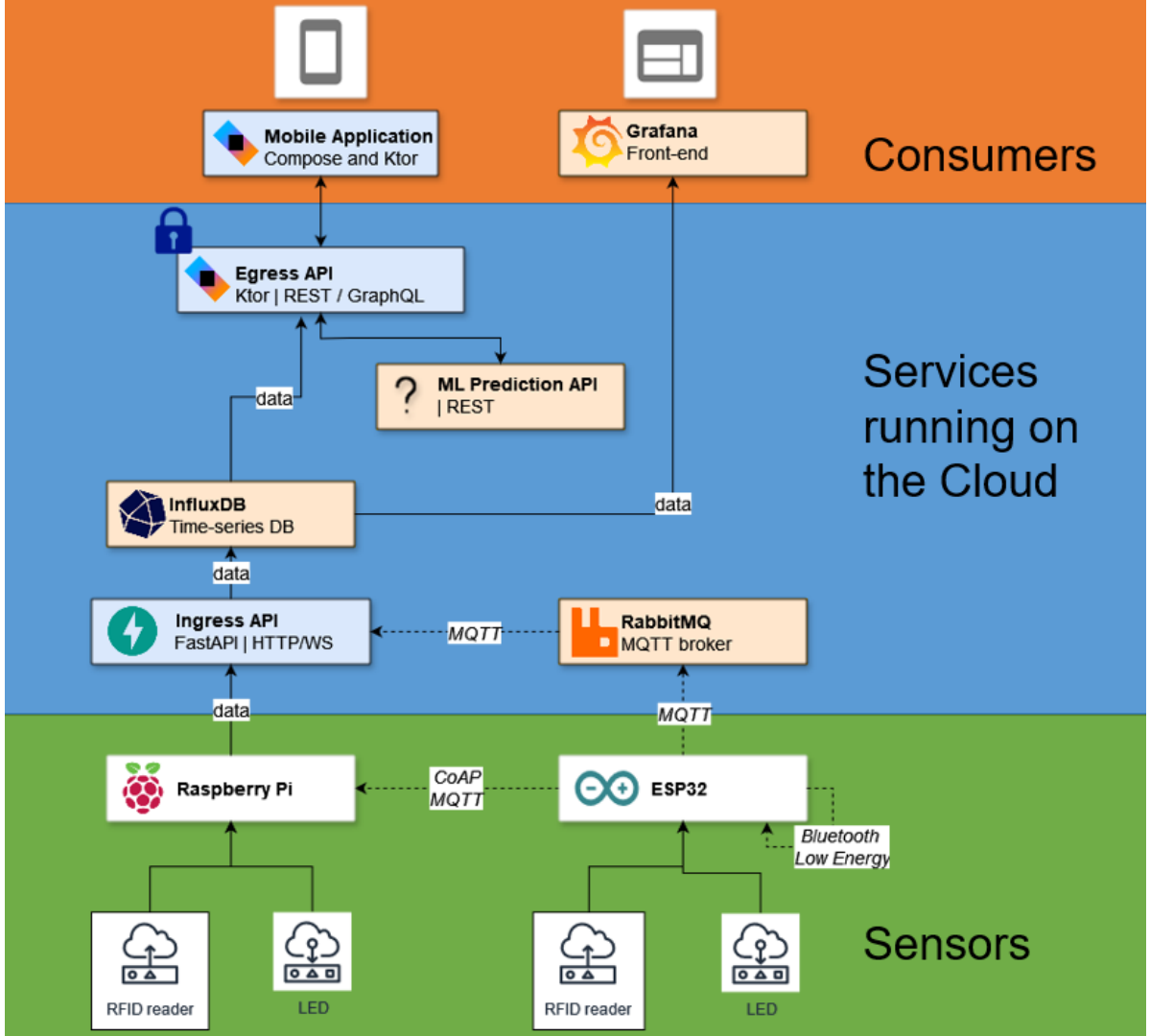
FACULTY OF ENGINEERING AND ARCHITECTURE

Lab 5 Egress – Part1: REST vs GraphQL

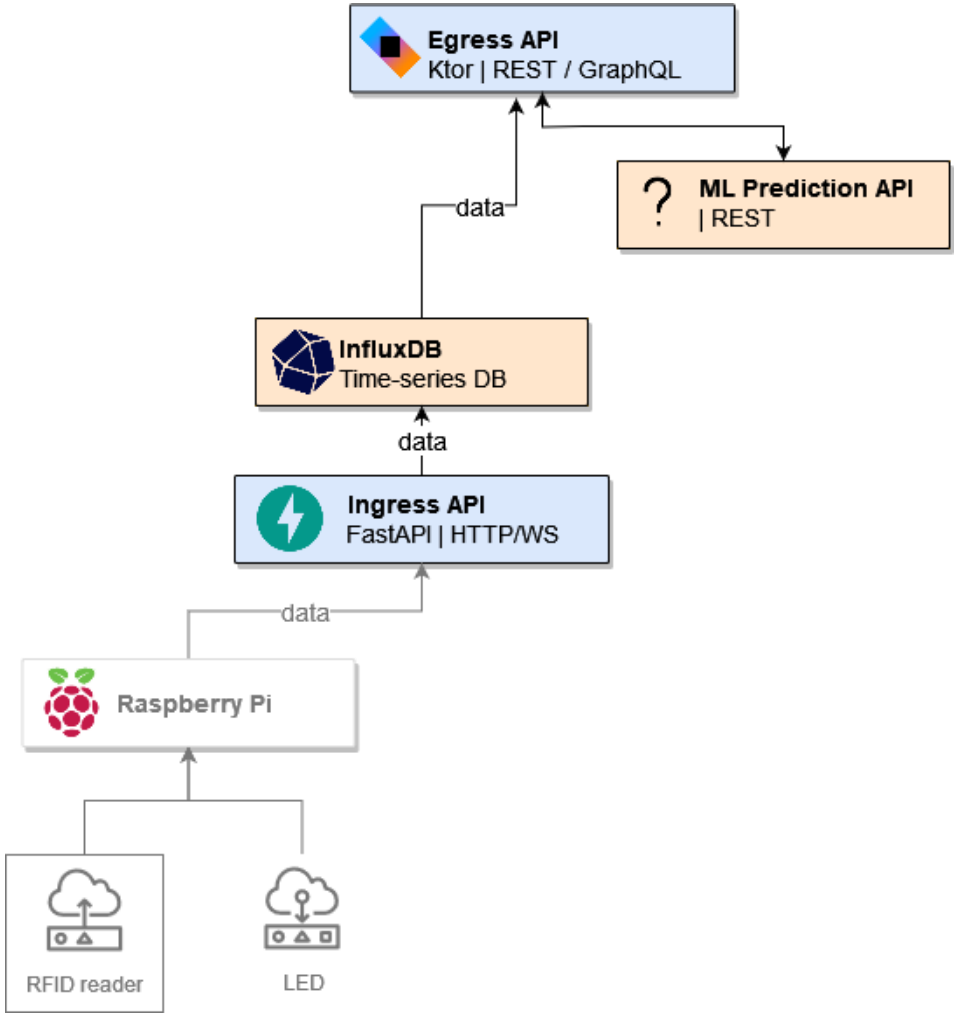
Cloud and mobile applications

GHENT UNIVERSITY



GHENT UNIVERSITY

Lab5



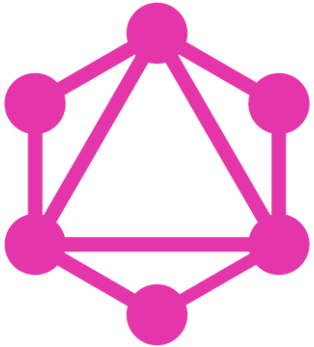
GHENT UNIVERSITY

Goals

1. Compose GraphQL queries to get specific data
2. Construct a GraphQL Schema in the GraphQL Schema Definition Language
3. Implement REST and GraphQL APIs
4. Illustrate the differences between REST and GraphQL
5. Interact with a third-party external API
6. Write asynchronous code (coroutines)

GraphQL

- Developed by Facebook
- Web APIs
- Structured queries
  - Returns structured responses
- Only select data that is needed
- Schema Definition Language (SDL)



{ REST }

GHENT UNIVERSITY

@Home: GraphiQL: Browser-based IDE for GraphQL querying

‹ Docs

⌘ K

Root

Fields

```
allFilms(
  after: String
  first: Int
  before: String
  last: Int
): FilmsConnection

film(
  id: ID
  filmID: ID
): Film

allPeople(
  after: String
  first: Int
  before: String
  last: Int
): PeopleConnection

person(
  id: ID
  personID: ID
): Person

allPlanets(
  after: String
  first: Int
```

```
1  {
2    person(id: "cGVvcGxlOjE=") {
3      name
4      birthYear
5      filmConnection {
6        films {
7          title
8        }
9      }
10     vehicleConnection {
11       vehicles {
12         name
13       }
14     }
15   }
16 }
17
18
```

Variables Headers

GraphiQL

```
"data": {
  "person": {
    "name": "Luke Skywalker",
    "birthYear": "19BBY",
    "filmConnection": {
      "films": [
        {
          "title": "A New Hope"
        },
        {
          "title": "The Empire Strikes Back"
        },
        {
          "title": "Return of the Jedi"
        },
        {
          "title": "Revenge of the Sith"
        }
      ]
    },
    "vehicleConnection": {
      "vehicles": [
        {
          "name": "Snowspeeder"
        },
        {
          "name": "Imperial Speeder Bike"
        }
```

GHENT UNIVERSITY

@Home: Get familiar with Kotlin

- Kotlin tutorial
- [understanding coroutines] → Needed for Task6
- Install IntelliJ IDEA
- Ktor documentation
  - Server framework
  - Routing
  - Call object
- Datetime
- Serialization

GHENT UNIVERSITY

In Lab: Tutorial

- Application Structure
  - Two modules:
    - REST
    - GraphQL
- Testing
  - Provided in `egress/src/test/kotlin/ApplicationTest.kt`

In Lab: Task 1: Implementing REST

- Implement four endpoints in REST
  - counts
  - ids
  - sources
  - attendance
- Only files that need to be adjusted are in :

  `egress/src/main/kotlin/server/modules/rest`

- Look at comments for expected format of return value!
- Queries are available in `kotlin/server/modules/Queries.kt`

GHENT UNIVERSITY

In Lab: Task 2: Implementing GraphQL

- Implement same endpoints as Task2 in GraphQL with QueryService
  - counts
  - ids
  - sources
  - attendance
- Only folder that contains files that needs to be filled in:

  `egress/src/main/kotlin/server/modules/graphql/`

- Look at comments for expected format of return value!
- Queries are available in `kotlin/server/modules/Queries.kt`

GHENT UNIVERSITY

In Lab: Task 3: Deploy Egress API to Cloud

- Add egress-api (online learning platform) to your existing microservice-oriented architecture
- Access egress service (locally) via port 8087
- Make sure you add your credentials in `egress/helm/values.yaml` and `egress/src/main/resources/influx2.properties`
  - Don't change values that are set up to point to environment variables

http://egress.<username>.cloudandmobile.ilabt.imec.be

NOTE: During 2nd practical session, **we will check egress in the cloud**, and REST and GraphQL endpoints should be implemented and deployed.

GHENT UNIVERSITY

In Lab: Optional Task 4: Differences GraphQL vs REST

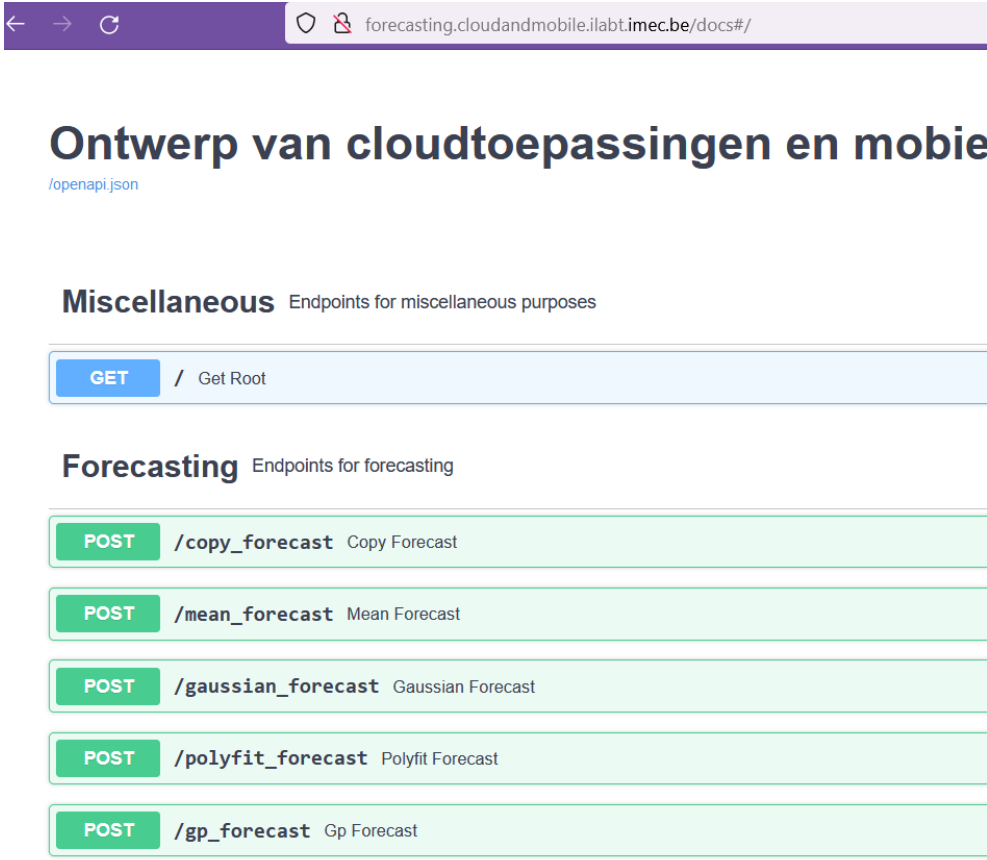
- Do not start with this task unless you have done all required tasks!

- Differences between GraphQL and REST
  - Timing
  - Response length
- Implement a separate client script
  - Executes queries on the same endpoint(s)
  - Check responses time-wise and length-wise
- Compare with your expectations and explain in report

GHENT UNIVERSITY

In Lab: Task 5: Obtaining forecasts

- Communicate with ML service that forecasts future behavior based on sensor data (count) from past
- GET from forecasting API to get predictions
- Only files that need to be adjusted are:
  `egress/src/main/kotlin/services/forecasting/Model.kt`
- **NOTE**: Tasks 5, 6, and 7 all have to be implemented before testing!



GHENT UNIVERSITY

In Lab: Task 6: Scheduling and caching forecasts

- If you only communicate with forecasting service when client asks, response time will be poor
- Forecasting job system to create new predictions over time and append to ServiceCache
- Uses Model.predict() from previous task
- Kotlin coroutines documentation is useful!
- Only files that need to be adjusted are:
  `egress/src/main/kotlin/services/forecasting/ServiceSyncJob.kt`

GHENT UNIVERSITY

In Lab: Task 7: GETting forecasts

- REST endpoint
- Expose forecast data to outside world
- Get data from ServiceCache (Task6)
- Only files that need to be adjusted are:
  `egress/src/main/kotlin/server/modules/rest/Forecast.kt`

- Test tasks 5, 6, and 7 after all are implemented

GHENT UNIVERSITY

Material to submit

- Preparation part at home: due **Thursday 27 March at 10:00**
  - Checklist on Ufora
- No report due until after Part2 finished. Part2 adds security to egress
- In total, Lab5 will take 3 weeks.
- During 2nd practical for this lab, we will check that tasks 1, 2, and 3 are completed
  - REST and GraphQL endpoints should be implemented and deployed to cloud

FACULTY OF ENGINEERING
AND ARCHITECTURE

Tom Windels
PhD student

Tom.Windels@ugent.be

Dr. Jennifer B. Sartor
Onderwijsbegeleider

Jennifer.Sartor@ugent.be

GHENT
UNIVERSITY