Question Answering and Chatbots

5.2th Practical exercise – Qanary Framework

Aleksandr Perevalov

aleksandr.perevalov@hs-anhalt.de

October 4, 2021

Hochschule Anhalt

Anhalt University of Applied Sciences

Plan for today

Plan for today

- Review the task for the Exercise 5.2;

Plan for today

- Review the task for the Exercise 5.2;
- Demo Session;

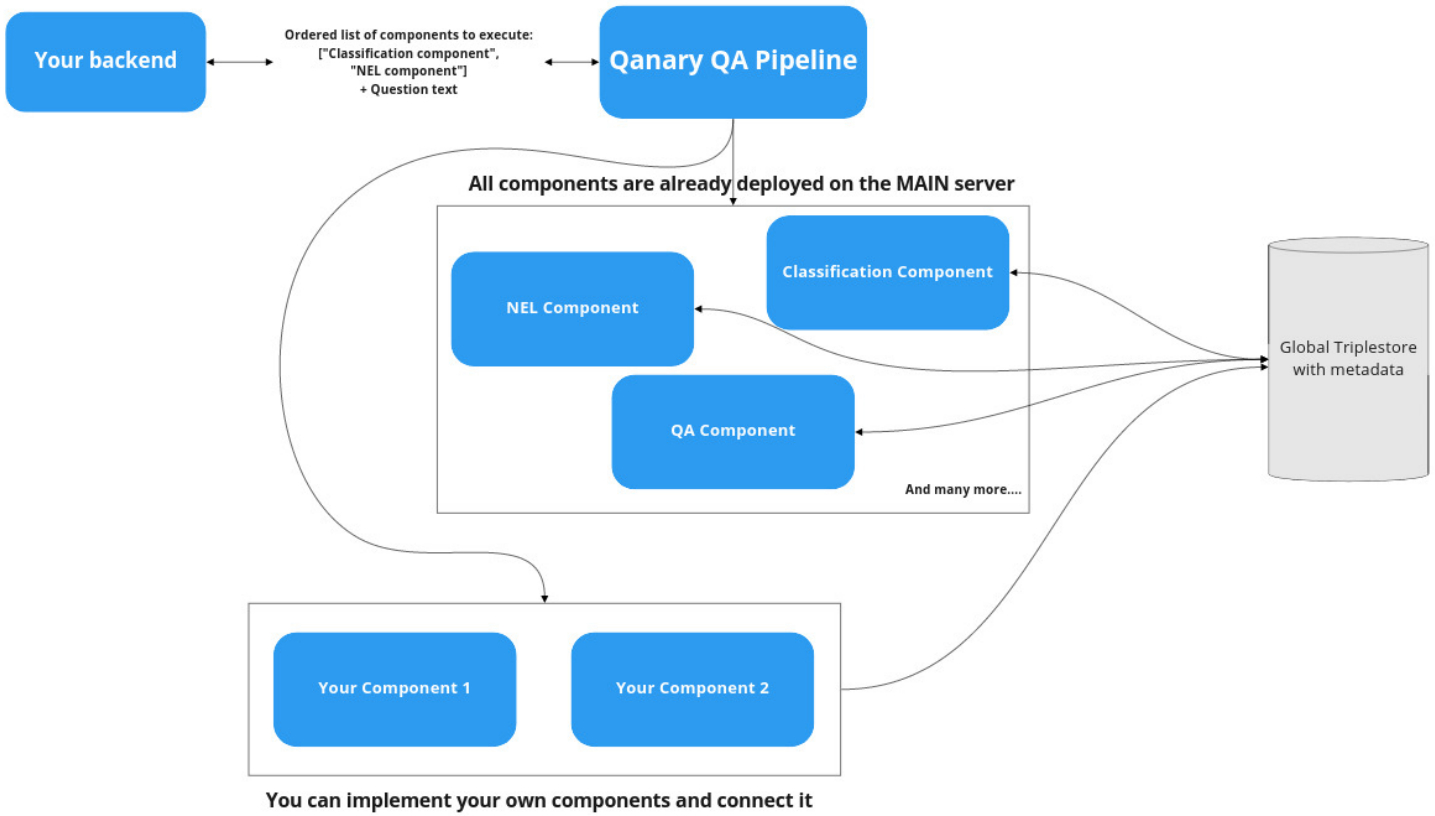
Plan for today

- Review the task for the Exercise 5.2;
- Demo Session;
- Questions;

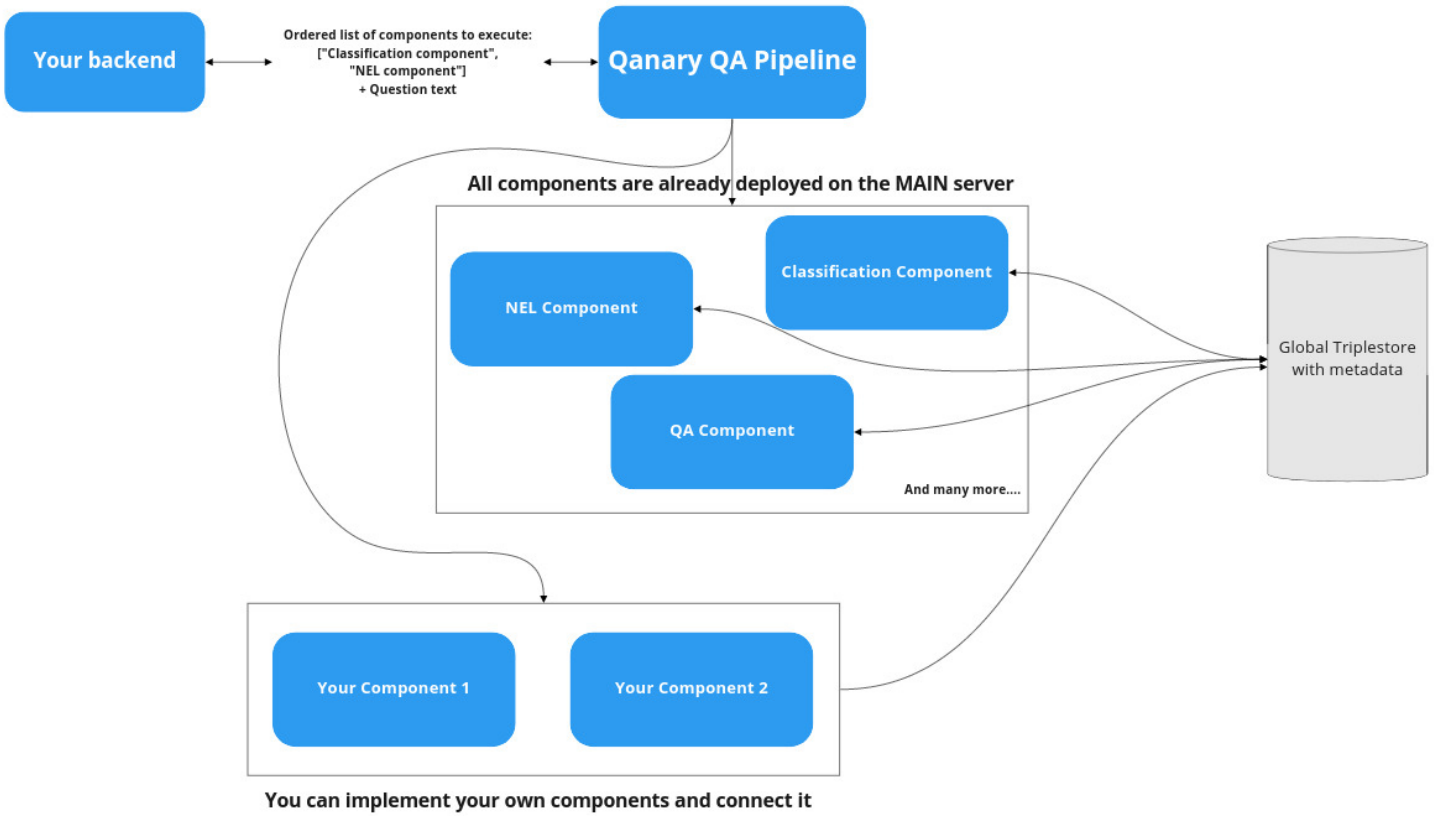
Plan for today

- Review the task for the Exercise 5.2;
- Demo Session;
- Questions;
- Introduction to the Exercise 6 (Tests).

Exercise 5.2 – the Task

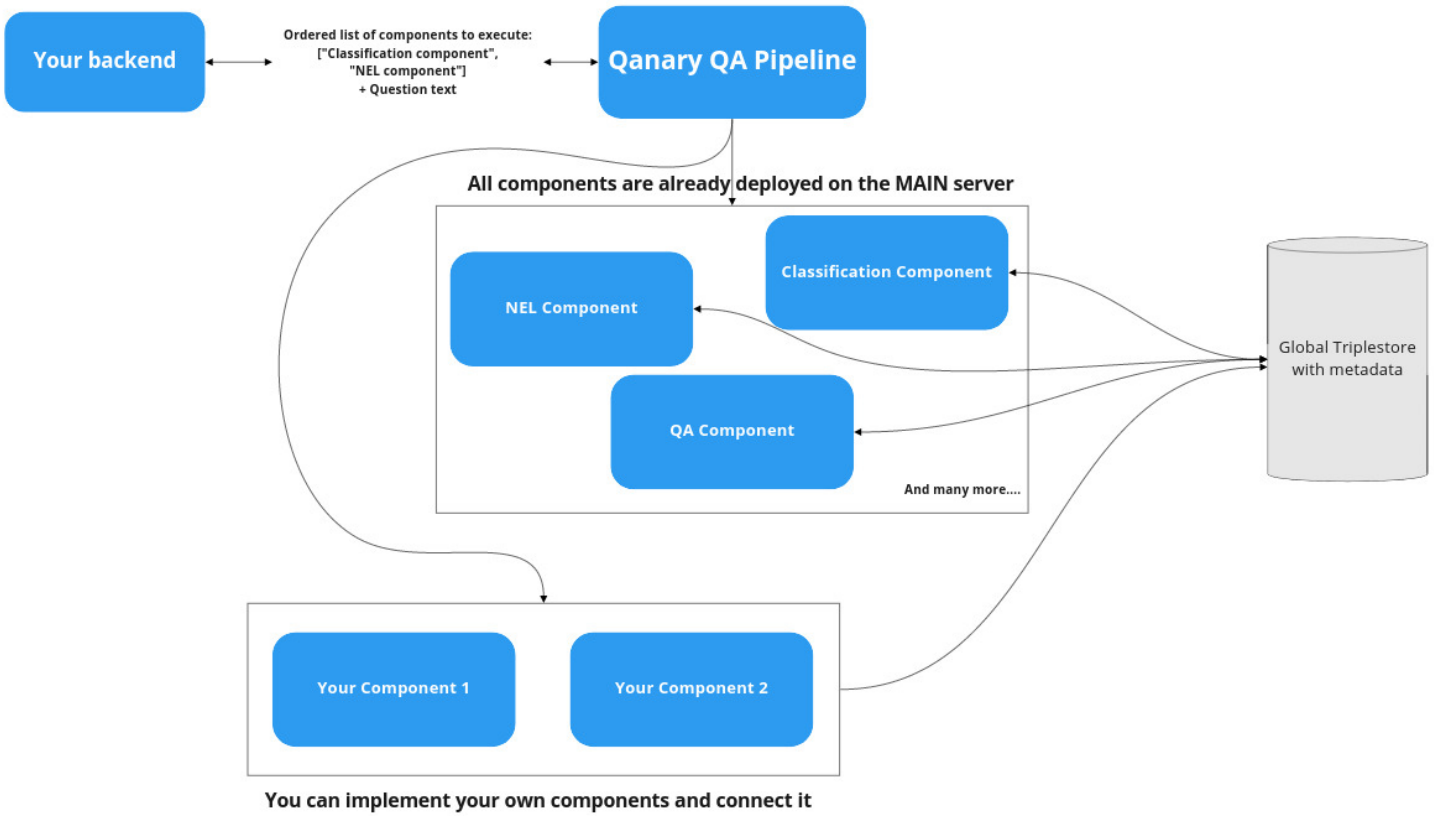


Exercise 5.2 – the Task



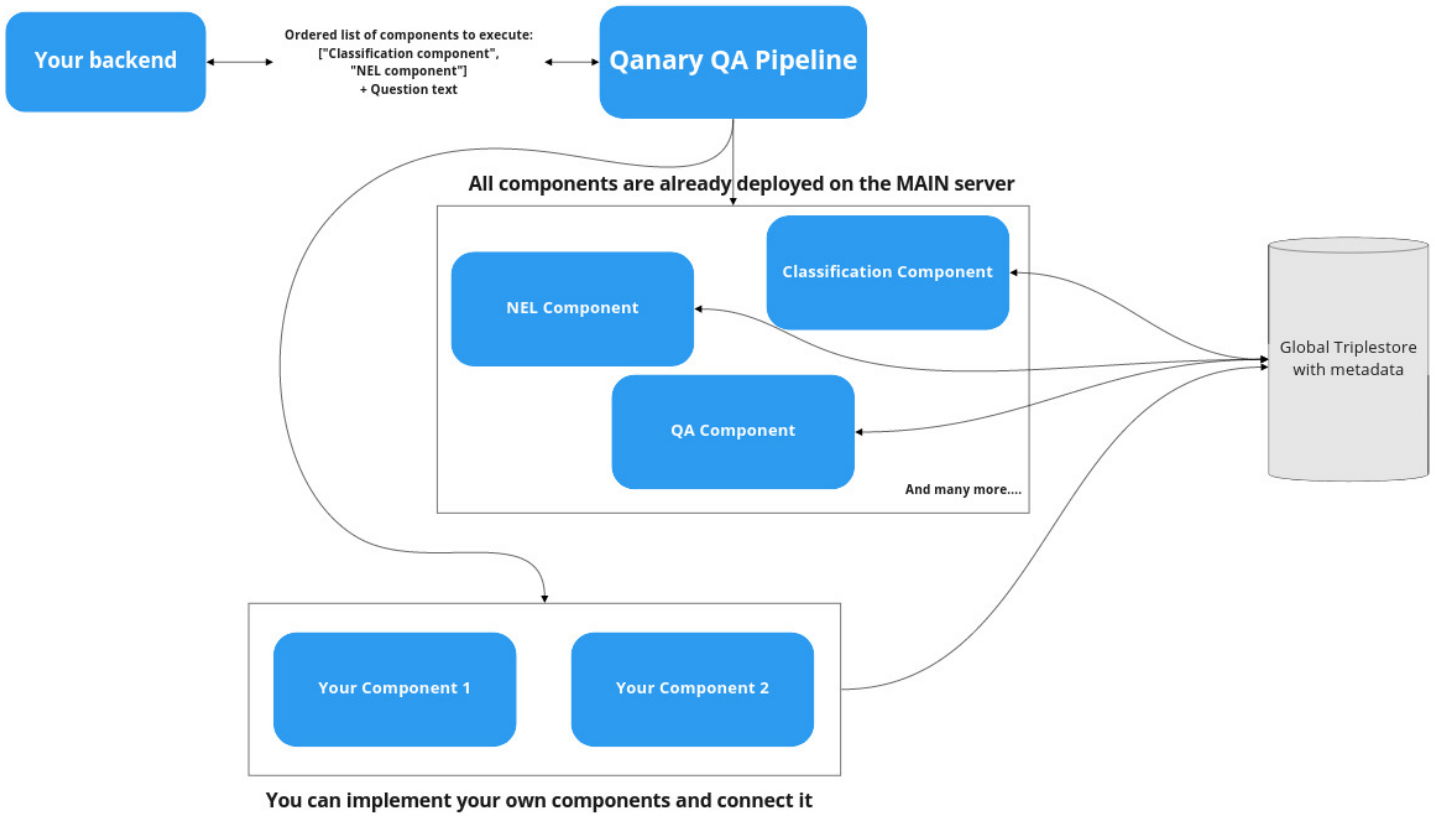
- Integrate the Qanary framework into the Backend;

Exercise 5.2 – the Task



- Integrate the Qanary framework into the Backend;
- Combine the Frontend and the Backend;

Exercise 5.2 – the Task



- Integrate the Qanary framework into the Backend;
- Combine the Frontend and the Backend;
- Answer control questions in README.

Exercise 5.2 – difficulties

Exercise 5.2 – difficulties

- How to expose multiple ports via `ngrok`?

Exercise 5.2 – difficulties

- How to expose multiple ports via `ngrok`?
- How to send a request to a component?

Any questions?

Let's start the demo!

Motivation for tests

Motivation for tests

- The error always exists;

Motivation for tests

- The error always exists;
- The error is always not where you are looking for it;

Motivation for tests

- The error always exists;
- The error is always not where you are looking for it;
- Debugging is the process of fixing errors, while coding is the process of creating errors;
- ...

Motivation for tests

- The error always exists;
- The error is always not where you are looking for it;
- Debugging is the process of fixing errors, while coding is the process of creating errors;
- ...
- It should be hard to write a test, but it should be easy to run a test;

Motivation for tests

- The error always exists;
- The error is always not where you are looking for it;
- Debugging is the process of fixing errors, while coding is the process of creating errors;
- ...
- It should be hard to write a test, but it should be easy to run a test;
- Print enough intermediate information to the console while the application is not in the release version.

Types of tests

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);
4. Sanity testing – verify only new functionality;

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);
4. Sanity testing – verify only new functionality;
5. Regression testing – check if changes affected existing functionality;

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);
4. Sanity testing – verify only new functionality;
5. Regression testing – check if changes affected existing functionality;
6. System testing – test end-to-end system;

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);
4. Sanity testing – verify only new functionality;
5. Regression testing – check if changes affected existing functionality;
6. System testing – test end-to-end system;
7. Performance testing – test speed, response time, resource consumption;

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);
4. Sanity testing – verify only new functionality;
5. Regression testing – check if changes affected existing functionality;
6. System testing – test end-to-end system;
7. Performance testing – test speed, response time, resource consumption;
8. Load (stress) testing – test sustainability under peak load;

Note: test types are NOT exclusive. Can be performed automatically or manually.

Types of tests

1. Unit testing – test individual functions (components);
2. Smoke testing – test very basic functionality on new build;
3. Integration testing – test interaction between components (scenario);
4. Sanity testing – verify only new functionality;
5. Regression testing – check if changes affected existing functionality;
6. System testing – test end-to-end system;
7. Performance testing – test speed, response time, resource consumption;
8. Load (stress) testing – test sustainability under peak load;
9. User acceptance testing – verification using user scenarios.

Note: test types are NOT exclusive. Can be performed automatically or manually.

Plan for the Exercise 6: Tests

- Question Answering quality: Precision@1;
- Components testing: check the correctness of the annotation;
- Models evaluation: run a test set through a model and check for target metrics;
- Unit tests: define test sets for your “utils”.

1. SPARQL;
2. Work with Natural Language (NER);
3. Question classification;
4. Back-end and Front-end;
5. **Simple QA system and Qanary Framework**;
6. Tests for QA system;
7. Deploying QA system;
8. ...