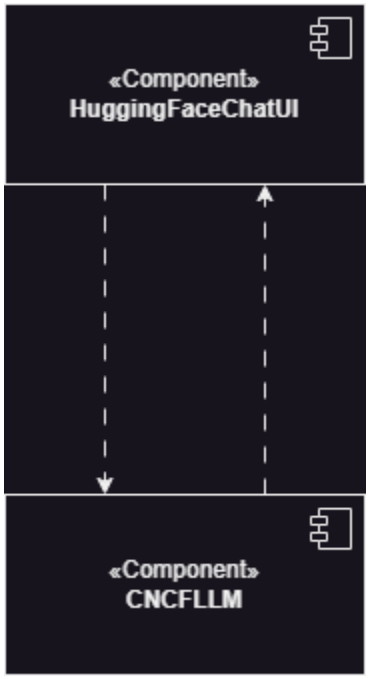
Software Architecture

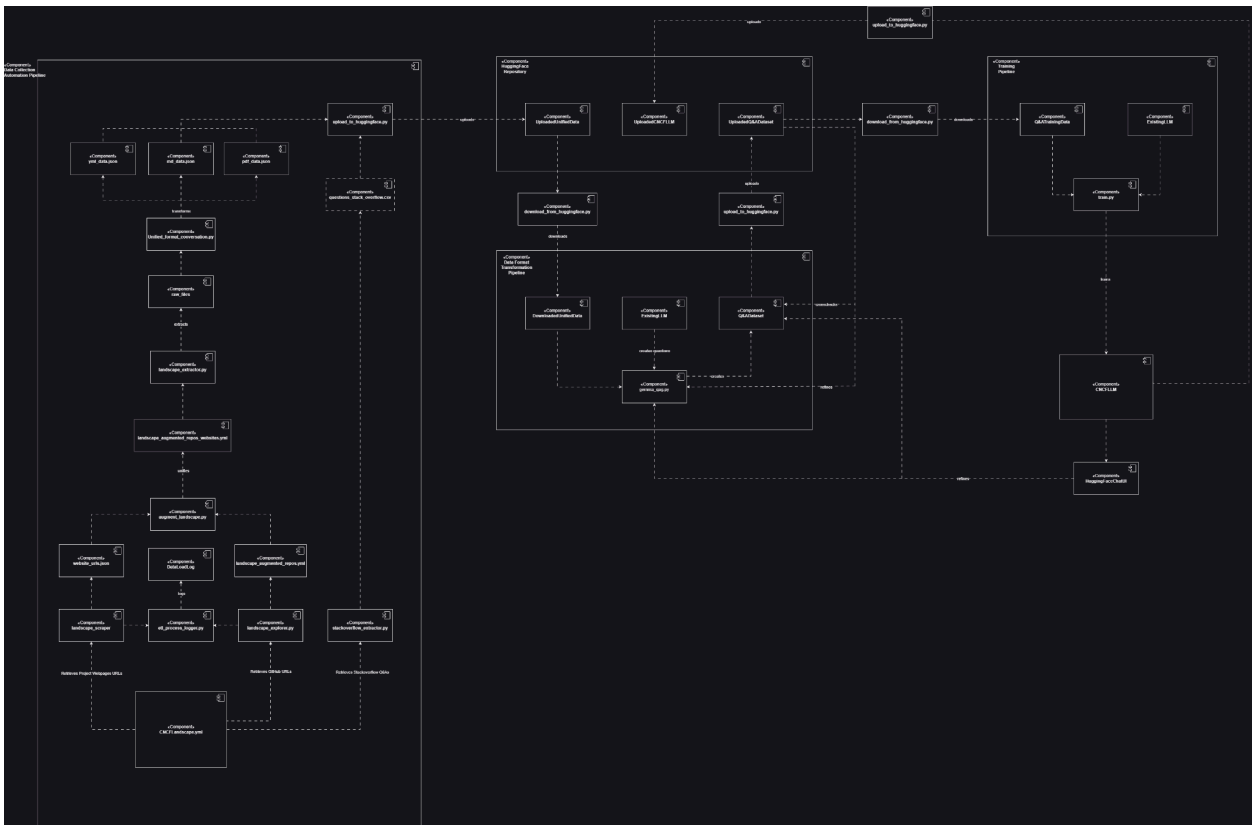
Runtime Components



- **HuggingFaceChatUI: Open-Source ChatUI for Large Language Models**
- **CNCFLLM: Existing Large Language Model tuned with Data from Cloud Native Computing Foundation projects' documentation**

The user can interact with the LLM via the ChatUI, asking CNCF related questions. The model then provides the user with elaborated answers.

Code Components



Summary

- **Data Collection Automation Pipeline: pipeline that is used to automatically extract, transform and load the CNCF data**
  - landscape_explorer.py: script to retrieve GitHub URLs to Markdown, pdf and YAML files of the projects listed in the CNCF landscape and store them in a YAML file
  - stackoverflow_extractor.py: script to extract Q&A pairs of the projects of the CNCF landscape from StackOverflow
  - landscape_scraper: component to retrieve Webpage URLs to pdfs, MD and Yaml files of the projects along with the documentation website urls listed in the CNCF landscape and store them in a json file
  - etl_process_logger.py: script to log the amount of loaded data and save the log in a DataLoadLog file
  - DataLoadLog: file containing the log of the amount of loaded data
  - website_urls.json: json file containing the retrieved URLs to the projects' documentation, pdfs, mds and yaml files
  - landscape_augmented_repos.yml: YAML file containing the retrieved URLs to Markdown, pdf and YAML files of the repositories of projects

- augment_landscape.python: script to augment previously created landscape_augmented_repos.yml with the scraped data from the websites
- landscape_augmented_repos_websites.yml: YAML file containing all relevant urls from the repositories and websites of the project
- landscape_extractor.py: script to extract Markdown, pdf and YAML files, and documentation of the projects and save them in an appropriate file format (pdf, yml, md)
- raw_files: folder where all extracted files from landscape_augmented_repos_websites.yml get downloaded
- Unified_format_conversation.py: script to unify the extracted data and save it in an appropriate file format
- md_data.json, pdf_data.json, yaml_data.json: files containing the unified extracted data
- questions_stack_overflow.csv: file containing all the extracted data from stackoverflow

- **HuggingFace Repository: repository where the unified data and the final Q&A dataset is hosted**
  - upload_to_huggingface.py: script for uploading the data to the huggingface repository
  - UploadedUnifiedData: file containing the unified extracted data uploaded to the huggingface repository
  - download_from_huggingface.py: script for downloading the data from the huggingface repository
  - UploadedQ&ADataset: file containing the Q&A dataset used for training

- **Data Format Transformation Pipeline: pipeline that is used to transform the unified data into a Q&A data format using an llm for creating question/answer pairs**
  - DownloadedUnifiedData: file containing the unified extracted data downloaded to the machine where the data format transformation pipeline is executed
  - ExistingLLM: LLM that is used as a base model for the CNCF model (e.g. Google GEMMA)
  - gemma_qag.py: script for creating the question/answer pair dataset that most llms expect for finetuning, currently using the gemma model for the Q&A generation
  - Q&ADataset: Dataset containing the question/answer pairs
  - train.py: script to train the exisiting llm model with the ner extracted data

- **Training Pipeline: pipeline that is used to train an existing LLM model with the extracted and transformed CNCF data**
  - Q&ATrainingData: file consisting of the question/answer pair data that is used for training an existing model
  - ExistingLLM: LLM that is used as a base model for the CNCF model (e.g. Google GEMMA)
  - train.py: script to train the exisiting llm model with the ner extracted data
- CNCFLLM: Model trained with extracted CNCF data that is able to answer CNCF related prompts
- HuggingFaceChatUI: User Interface that uses the CNCF LLM and allows for user interaction with the model

So far, our project's code structure consists of three main pipelines, the data collection automation pipeline which combines all procedures needed to automatically extract the CNCF data from the internet, the Data Format Transformation Pipeline which transforms the unified data into a dataset consisting of question/answer pairs needed for finetuning existing llms, and the training pipeline which tunes an exisiting LLM with the extracted CNCF q&a data so that it's able to elaborately answer queries related to CNCF projects. The resulting model is then loaded into the open-source HuggingFaceChatUI in order to provide a user interface that allows for user interaction with the model. The unified data, the Q&A dataset and the final CNCFLLM are uploaded to a huggingface repository. For the remainder of this project, the Q&A data is crosschecked in order to improve it and to refine the gemma_qag.py. The same evaluation is done using the current iteration of the CNCFLLM.

Architecture Choices

Programming language

We use python for gathering and extracting the CNCF landscape data as it's a modern language that provides us with numerous packages we can use for a fast and easy ETL process

Machine learning library

We use Pytorch for training the existing model as it provides us with an easy to use interface and is well-documented

AI Model

Google GEMMA 7B is our first choice for an existing LLM to build upon, as it's a performant model and has an open license, however we'll might try out other alternatives if they fit better later on. If needed, the model might be changed towards a bigger existing version. We are currently using it also to generate Q&A pairs from the unified data.

User Interface

We decided on the HuggingFaceChatUI as our Chat User Interface as it's open source, easy-to-use, and popular. As our main work consists of creating a well-functioning model,

we decided to use an exisiting AI chat interface. As it's open source, we can add functionalities if we'll need them in the future.

Technology Stack

Frontend

- HuggingFaceChatUI: TypeScript, Svelte

Backend

- Python
- Pytorch
- Python Mock
- Python Logging