BuggySoft Bug Tracker

Final Presentation

TABLE OF CONTENTS

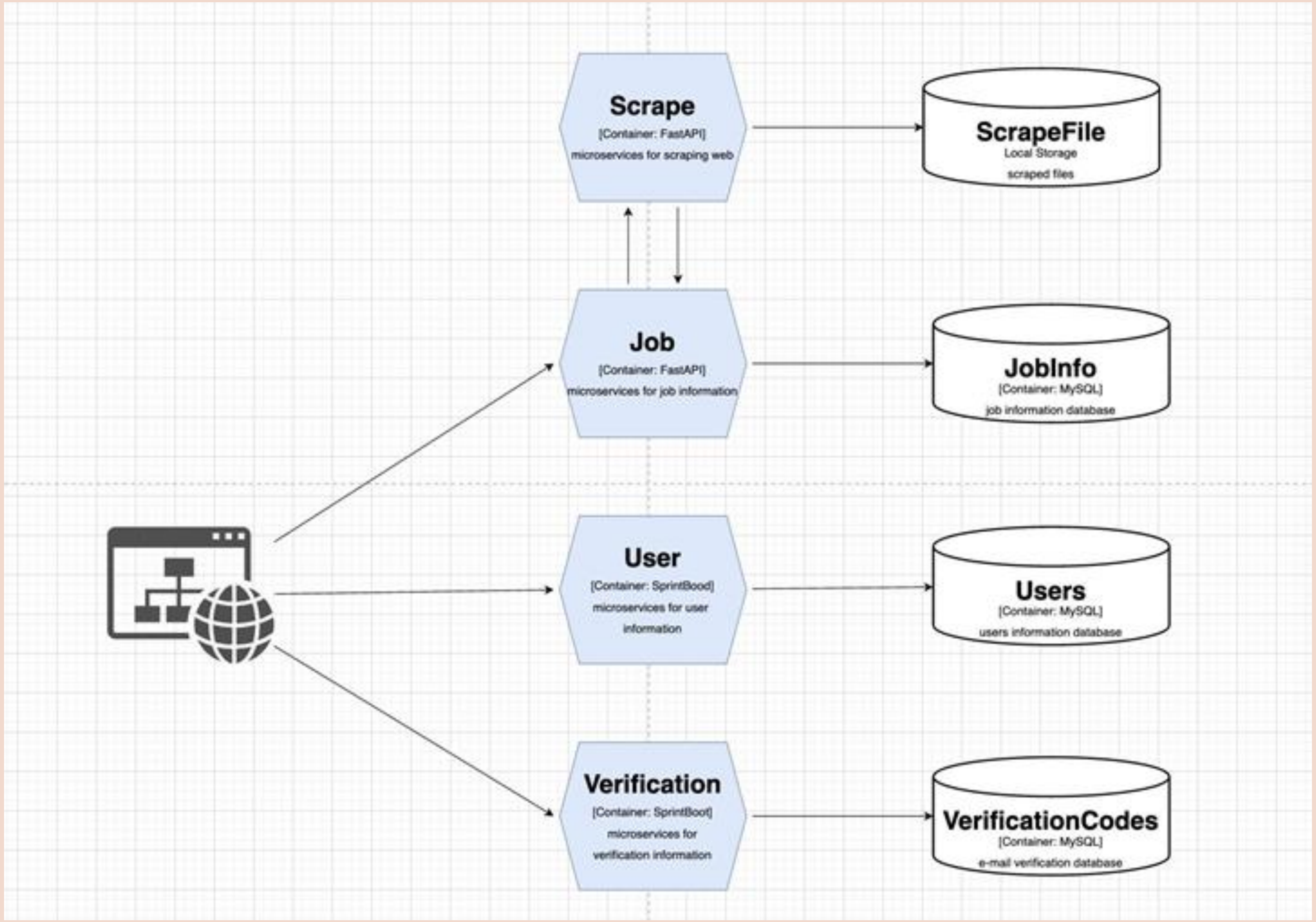
01

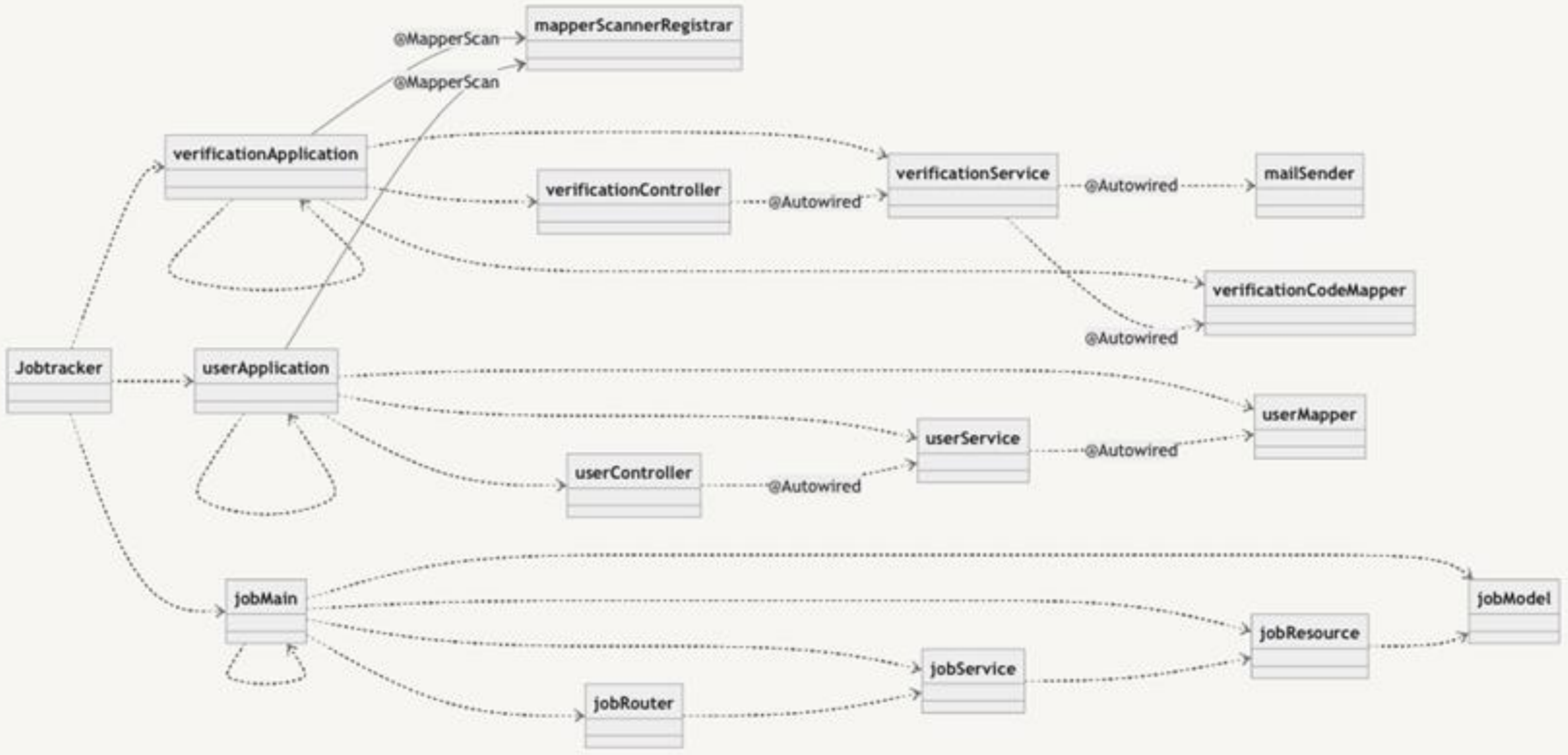
Project Overview

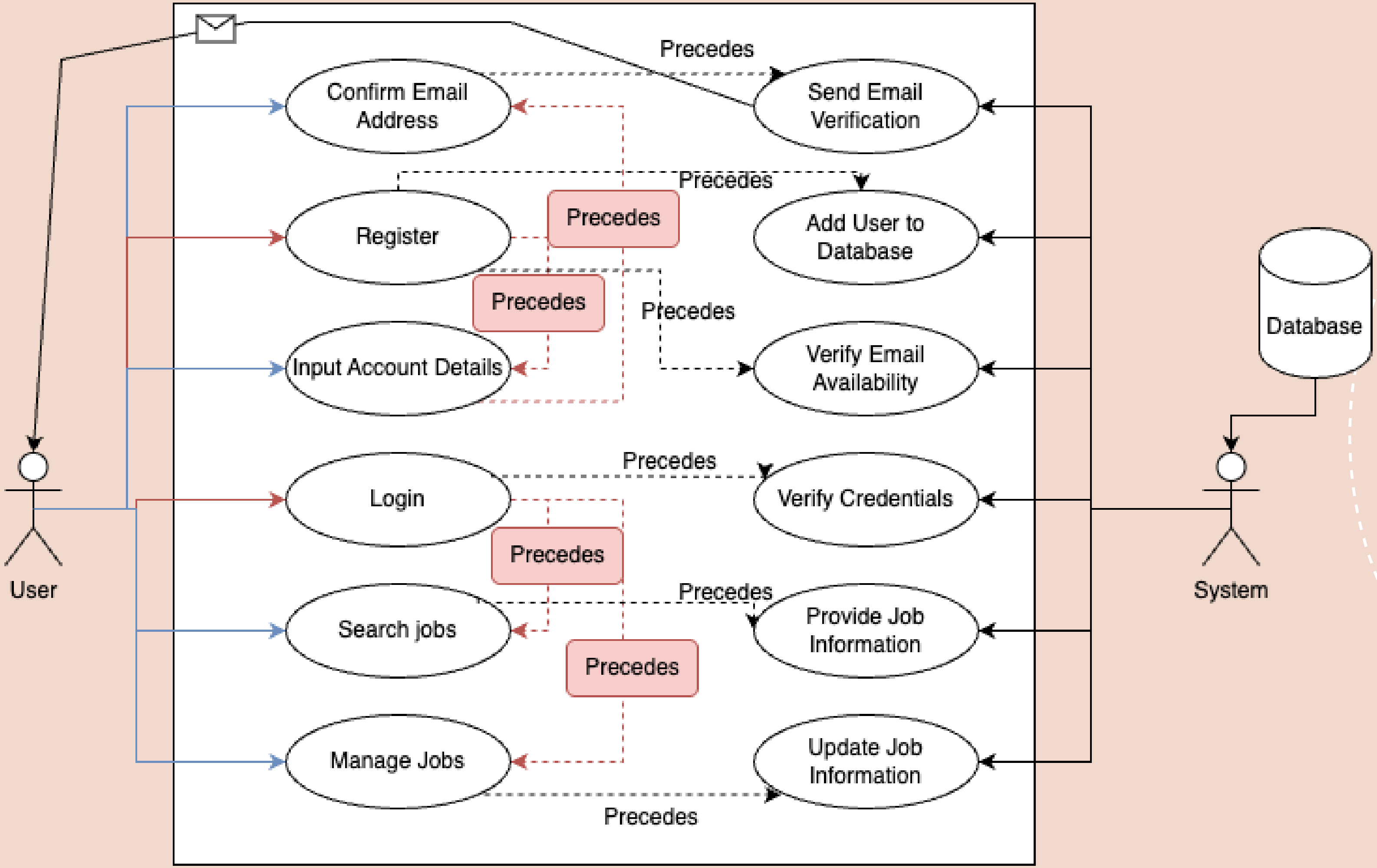
What is BuggySoft?

What is BuggySoft?

- Job search is hard
    - As new students in the CS field, we often have to apply to hundreds of jobs before securing one position
- Web application designed to help job seekers efficiently manage their job search
- Core features:
    - User authentication and verification through secure service
    - Job scraping – user simply inputs link to job posting in our app, we retrieve the information for you
    - Search through our database for a specific job by job ID
    - Keep everything in one organized location


Scrape
[Container: FastAPI]
microservices for scraping web
ScrapeFile
Local Storage
scraped files
Job
[Container: FastAPI]
microservices for job information
JobInfo
[Container: MySQL]
job information database
User
[Container: SprintBood]
microservices for user information
Users
[Container: MySQL]
users information database
Verification
[Container: SprintBoot]
microservices for verification information
VerificationCodes
[Container: MySQL]
e-mail verification database


mapperScannerRegistrar
@MapperScan
@MapperScan
verificationApplication
verificationController
@Autowired
verificationService
@Autowired
mailSender
verificationCodeMapper
@Autowired
Jobtracker
userApplication
userMapper
userService
@Autowired
userController
@Autowired
jobMain
jobModel
jobResource
jobService
jobRouter


Precedes
Confirm Email Address
Send Email Verification
Precedes
Register
Precedes
Add User to Database
Precedes
Precedes
Verify Email Availability
Input Account Details
Database
Precedes
Login
Verify Credentials
Precedes
User
System
Precedes
Search jobs
Provide Job Information
Precedes
Manage Jobs
Update Job Information
Precedes

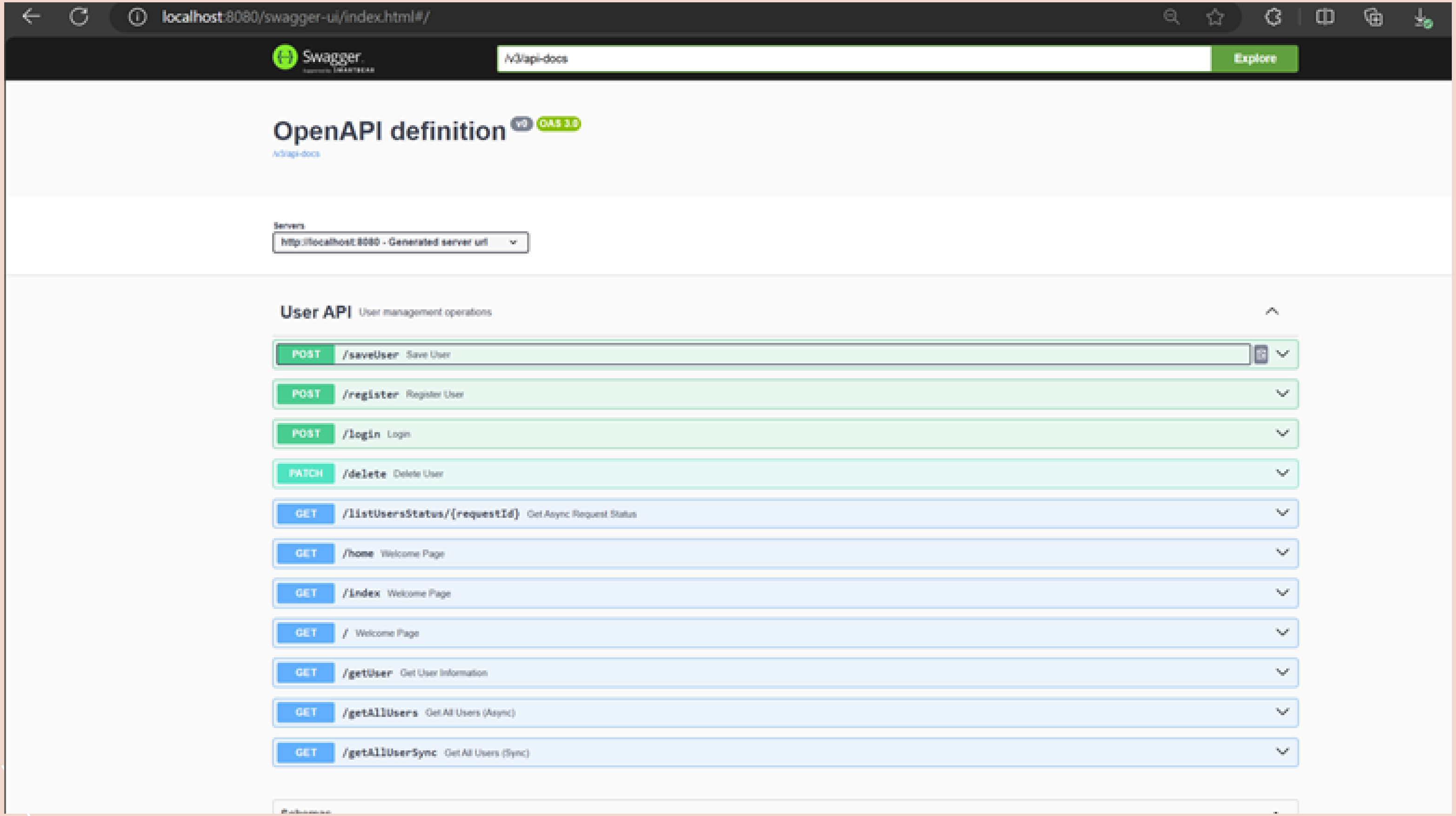
02

Services

The Parts

User Service

- User storage service for BuggySoft Job Tracker
- Built in Java 17, using SpringBoot
- Microservice architecture with RESTful endpoints for modular functionality
- Responsible for user authentication, management, and storage
    - Specific functionality:
        - User registration, login, deletion, storage in backend database
        - Bulk operations for retrieving list of all users
            - Status monitoring for asynchronous requests involving bulk operations



localhost:8080/swagger-ui/index.html#/

Swagger

/v3/api-docs

Explore

OpenAPI definition v0 OAS 3.0

/v3/api-docs

Servers

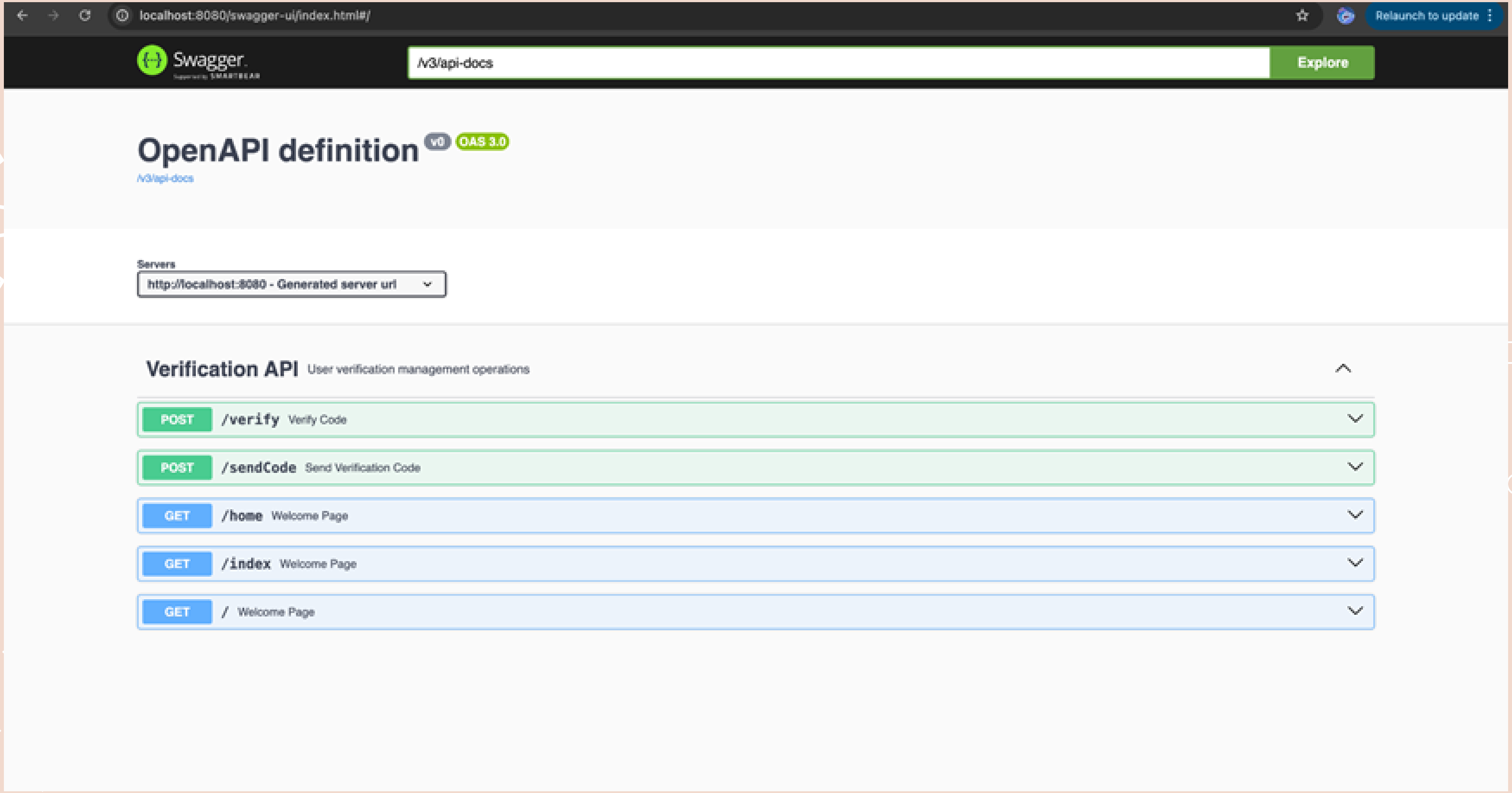
http://localhost:8080 - Generated server url

User API User management operations

POST	/saveUser	Save User
POST	/register	Register User
POST	/login	Login
PATCH	/delete	Delete User
GET	/listUsersStatus/{requestId}	Get Async Request Status
GET	/home	Welcome Page
GET	/index	Welcome Page
GET	/	Welcome Page
GET	/getUser	Get User Information
GET	/getAllUsers	Get All Users (Async)
GET	/getAllUserSync	Get All Users (Sync)

Verification Service

- Email verification service for BuggySoft Job Tracker
- Built in Java 17, using SpringBoot
- Microservice architecture with RESTful endpoints for modular functionality
- Responsible for authenticating user login attempts
    - Specific functionality:
        - Ensure secure account activation and authentication
        - Send and receive verification codes, sent to user email upon registration attempt



OpenAPI definition v0 OAS 3.0

/v3/api-docs

Servers

http://localhost:8080 - Generated server url

Verification API User verification management operations

POST /verify Verify Code

POST /sendCode Send Verification Code

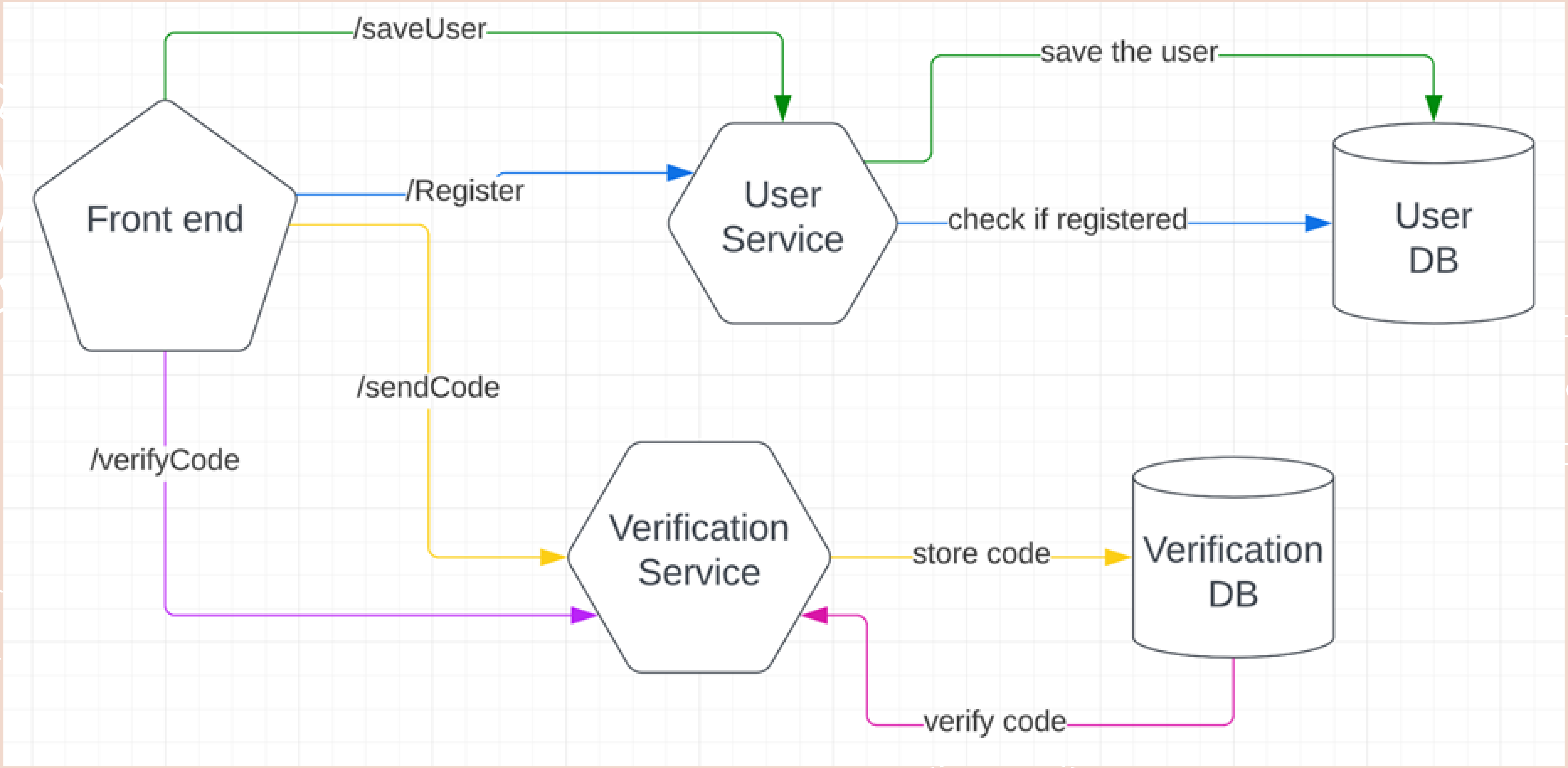
GET /home Welcome Page

GET /index Welcome Page

GET / Welcome Page

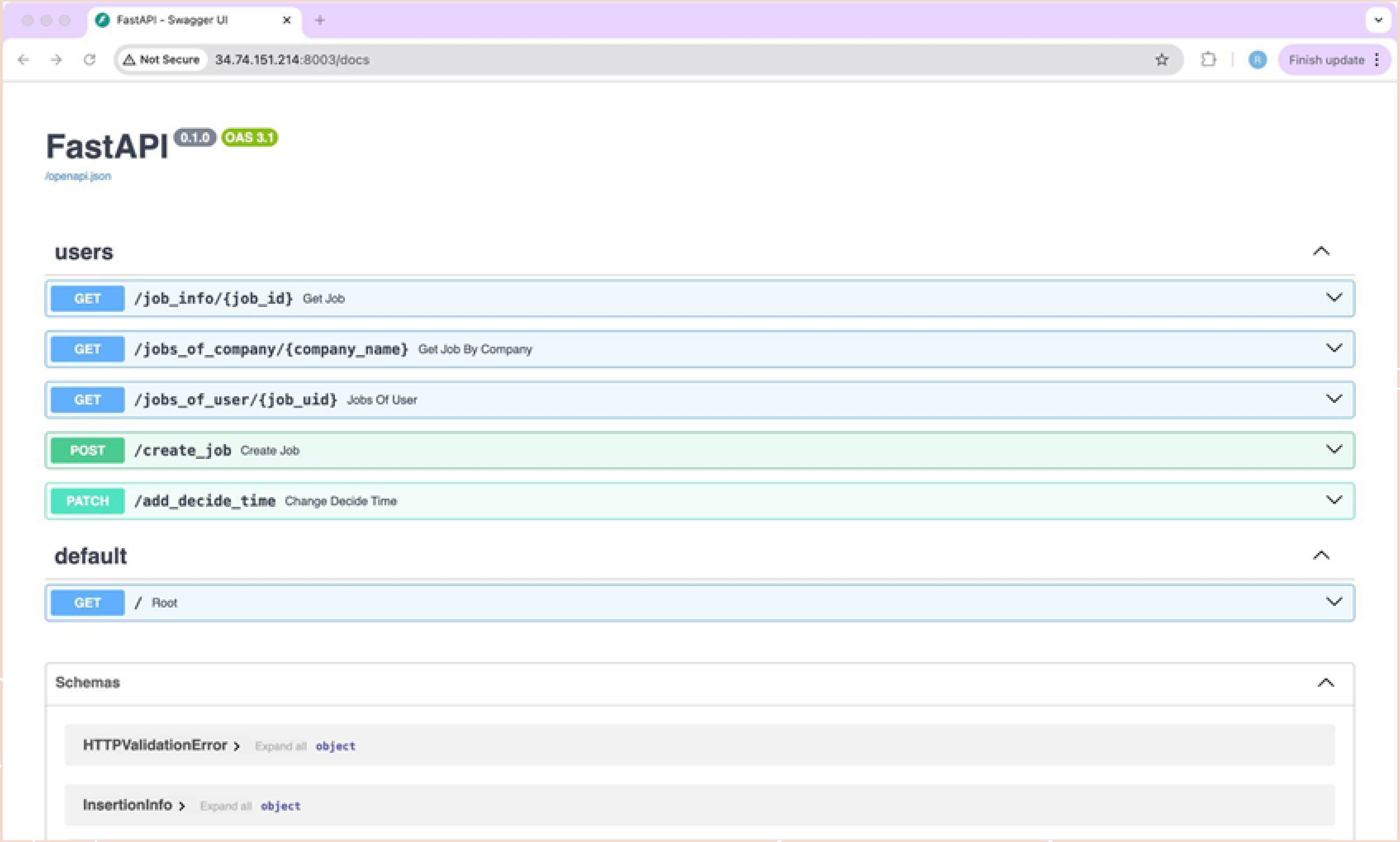
Communication (User & Verification)

- Frontend UI communicates with registration endpoint on user service
- Upon receiving response from service, UI then communicates with verification service to send a code to the newly registered email, prompting the user to enter the code
- Only upon receipt of correct code will the UI call the /saveUser endpoint (hidden endpoint) to actually save the user


/saveUser
save the user
Front end
/Register
User Service
check if registered
User DB
/sendCode
/verifyCode
Verification Service
store code
Verification DB
verify code

Job Service

- Service for user to store or retrieve the job they are interested in
- Built in Python 3.11.2, using FastAPI
- A composite microservice (interact with Scrape Service) with RESTful endpoints
- Responsible for storing all job related information for users
    - Specific functionality:
        - Store job and all related information that the user is interested in;
        - Retrieve all jobs and their related information by searching keywords



FastAPI - Swagger UI

Not Secure 34.74.151.214:8003/docs

Finish update

FastAPI 0.1.0 OAS 3.1

/openapi.json

users

GET /job_info/{job_id} Get Job

GET /jobs_of_company/{company_name} Get Job By Company

GET /jobs_of_user/{job_uid} Jobs Of User

POST /create_job Create Job

PATCH /add_decide_time Change Decide Time

default

GET / Root

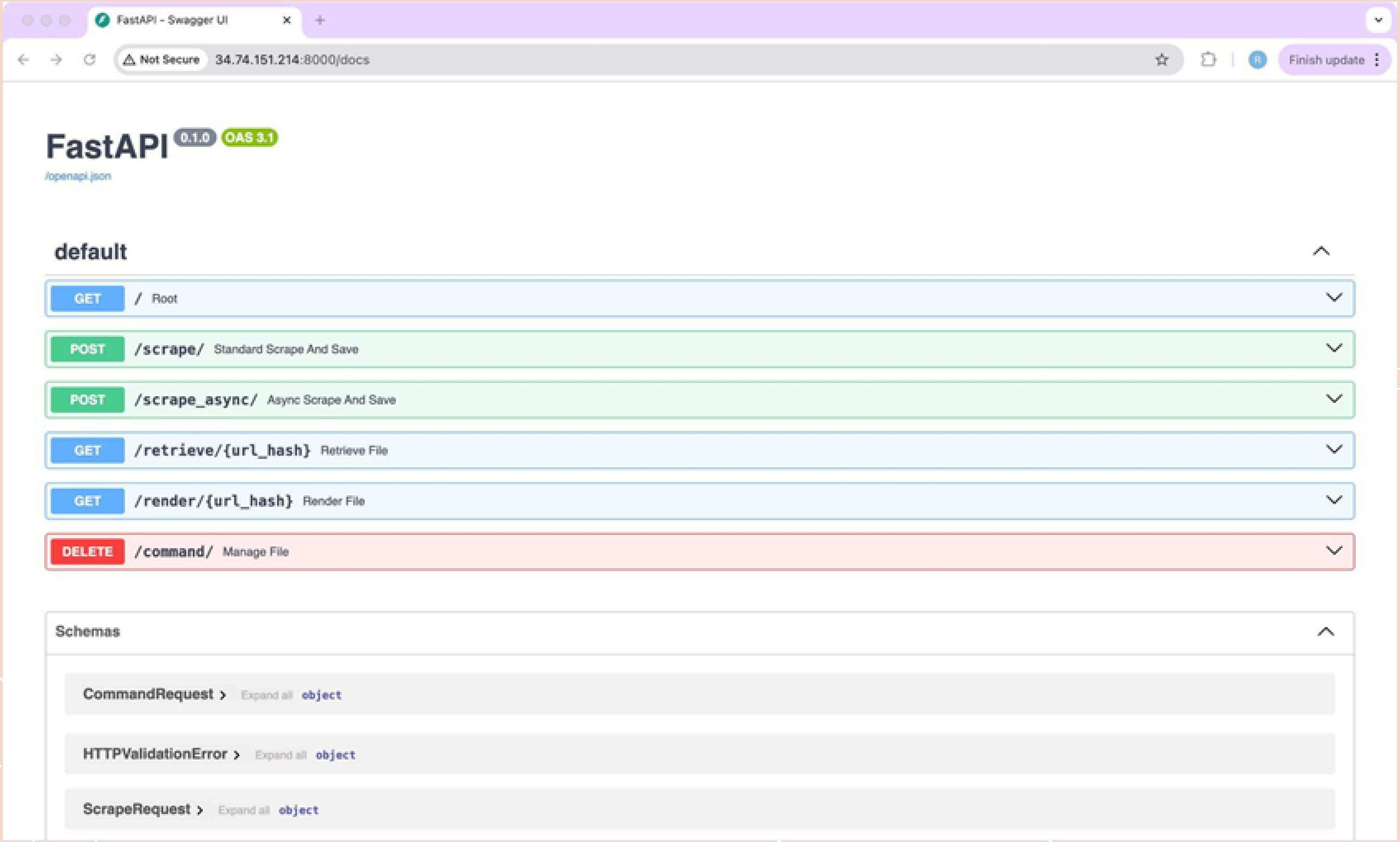
Schemas

HTTPValidationError > Expand all object

InsertionInfo > Expand all object

Scrape Service

- Service to scrape the website for job related information given the URL
- Built in Python 3.11.2, using FastAPI
- A composite microservice (interact with Job Service) with RESTful endpoints
- Responsible for scraping the web page and retrieving the scraped file
    - Specific functionality:
        - Scrape the given URL webpage and return a handle to the scraped data;
        - Retrieve the scraped file for the given hash


FastAPI - Swagger UI

Not Secure 34.74.151.214:8000/docs

Finish update

FastAPI 0.1.0 OAS 3.1

/openapi.json

default

GET / Root

POST /scrape/ Standard Scrape And Save

POST /scrape_async/ Async Scrape And Save

GET /retrieve/{url_hash} Retrieve File

GET /render/{url_hash} Render File

DELETE /command/ Manage File

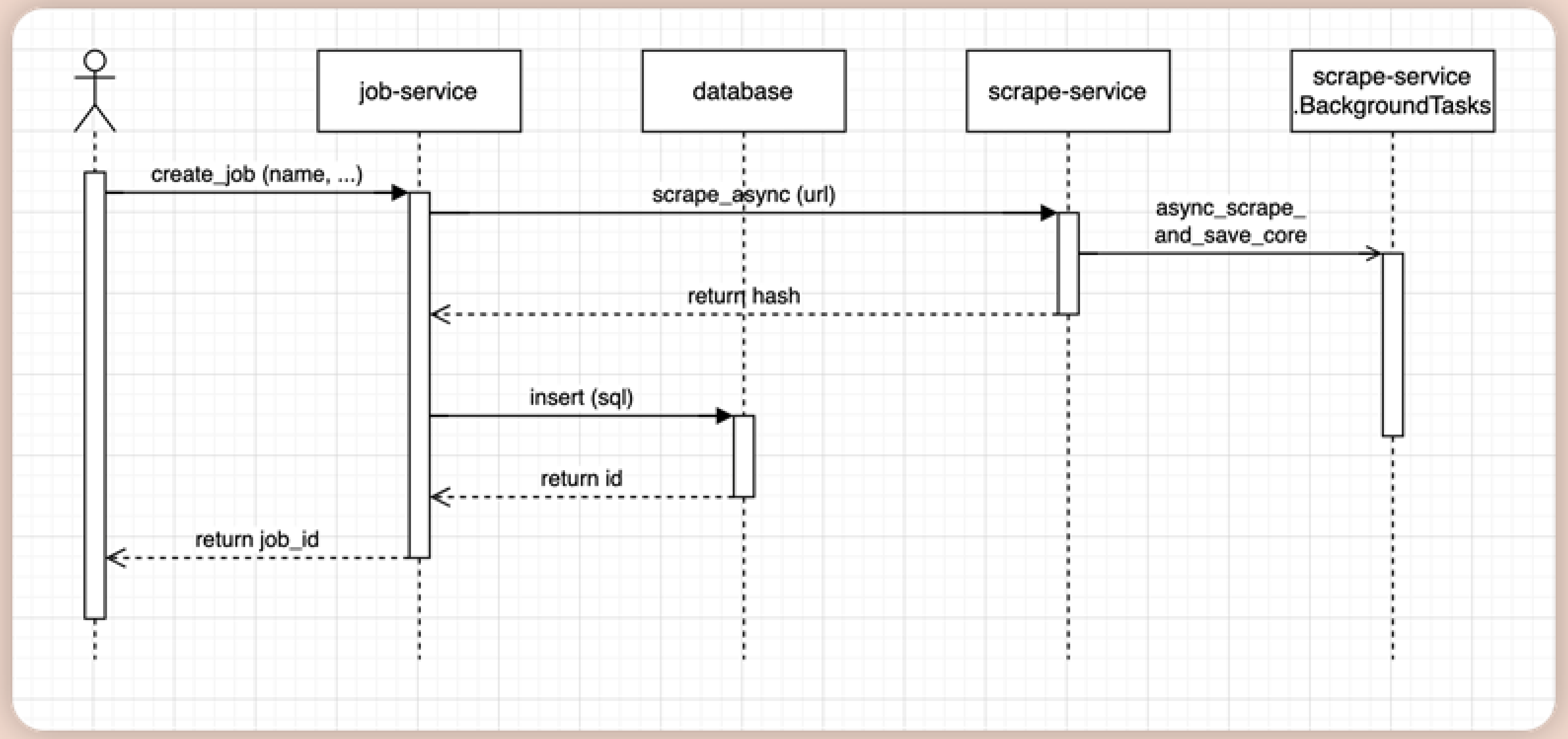
Schemas

CommandRequest › Expand all object

HTTPValidationError › Expand all object

ScrapeRequest › Expand all object

Communication (Job & Scrape)



03

Demo

The Runthrough

Demo

04

Contributions

The Teams

Division of Labor

- Three Teams
  - “Java” team – responsible for implementation + development of user and verification services
  - “Python” team – responsible for implementation + development of job and scraping services
  - UI team – responsible for frontend UI design and logic development
- Meeting coordination and milestone management
  - Jiakai Xu + Suwei Ma

Contributions

- Java Group
  - Suwei Ma: Implementation of user and verification services, documentation, completion of REST interface for user service, 202 Accepted (asynchronous and synchronous), OpenAPI documentation, Sprint report organization
  - Avery Fan: Implementation of user and verification services, cloud database setup and deployment on Google CloudSQL, physical model diagram, logging logic for services, links section, pagination, JWT tokens
  - Michael Wang: Assistance with user and verification service implementation, documentation, 201 Created + helper functions for user service, implemented GET on link header for created resource
- Python Group
  - Jiakai Xu: Design and implementation of Scrape service; deploy Job and Scrape service on the server; assist development of Job service and middlewares; manage GitHub Organization
  - Ruizhe Fu: Initial and basic implementation for Job services; implement the connection between the Job service and Scrape service; implement Patch ; documentation; presentation
- UI Group
  - Haiyue Zhang: UI/UX design for Landing, Login, Verification Pages, Utilize user and verify service APIs to enable functionality, Deploy frontend on AWS EC2
  - Tianle Zhou: UI/UX design for Job services, Create Account Pages, Utilize job and scrape service APIs to implement functionality, Demo Video record

What Next

- OAuthentication
  - Google?
- Use of external cloud services
- CI/CD
- Correlation ID/Propagation
- Service choreography/orchestration
- Cloud blobstore for UI
- Minor tweaks