



HelpMe&You



PRESENTATO DA:
Anna Fontana
Gianni Molinari
Fernando Serrano

14/11/2023

Indice

Introduzione



Project Overview



Workflow Overview: Trello

Indice

Backend

- 3 Architettura a microservizi
- 4 Microservizi
- 5 Strumenti di supporto:
 - Api Gateway
 - RabbitMQ
 - Eureka Server
- 6 Dataflow
- 7 Funzionalità aggiuntive:
 - File multimediali

Indice

Frontend

8

Vue

9

Login con Google

10

Funzionalità aggiuntive:

- Gestione della sessione utente
- Gestione del routing tra le pagine
- Gestione delle richieste scadute

11

Demo

Indice

Conclusione

- 12 Docker
- 13 Kubernetes
- 14 GitHub e README
- 15 Sviluppi futuri

Indice

Introduzione



Project Overview



Workflow Overview: Trello

Project Overview

Idea - Report ISTAT 2022

- “2021: sono in condizione di povertà assoluta poco più di 1,9 milioni di famiglie, circa 5,6 milioni di individui. La povertà assoluta conferma i massimi storici toccati nel 2020, anno della pandemia di Covid-19”
- “1,3 milioni di persone non hanno nessuno su cui poter contare e un cittadino su tre si sente più solo di prima della pandemia. Le persone che già soffrivano di solitudine sono le più colpite, ciò ha inasprito condizioni di maggiore vulnerabilità e pregresse criticità”

Project Overview

Features

Web app per Utente, Magazzino e Amministratore:

- Consentire donazioni per l'associazione di volontariato
- Pubblicare e accettare richieste di aiuto
- Acquistare e gestire materiale per il magazzino
- Inviare e gestire segnalazioni di richieste di aiuto

e molto altro nella demo...

Indice

Introduzione

1

Project Overview

2

Workflow Overview: Trello

Project TAAS ★ Visibile allo Spazio di lavoro Bacheca

Da fare

- RICORDA (1/1)
- TODO (24/24)

In esecuzione

- Class Diagram (26 ott 2022)
- Spikes to be done (0/6)
- Motivations
- Project Goals
- Project NoGoals
- Initial project plan summary
- Similar Applications
- User stories (21 ott 2022, 10/10, FS, GM, AF)
- Use Case
- Scenarios
- CRC Cards - first draft (24 ott 2022)

Fatto

- DOMANDE (1/10)
- Account + segnalazioni (@giannimolinari1)
- Magazzino (@fernadoserrano75)
- Richieste di aiuto (@annaf147)
- Registrazione con Google (@annaf147)
- Collegare con rabbit magazzino e richieste di aiuto
- Dockerizzare i microservizi
- Dockerizzare frontend

Banca @giannimolinari1

⌚ Mar 1

Account + segnalazioni
@giannimolinari1

⌚ Apr 1

Magazzino @fernadoserrano75

⌚ Apr 1

Richieste di aiuto @annaf147

⌚ Apr 1

Registrazione con Google
@annaf147

⌚ May 1

Collegare con rabbit magazzino e richieste di aiuto

⌚ May 1

Dockerizzare i microservizi

⌚ Jun 1

Inserire su un pod di kubernetes tutti i container dei microservizi @giannimolinari1

⌚ Jun 15

Iniziare a settare il tutto @fernadoserrano75

⌚ Jul 26

Google da frontend @annaf147

⌚ Aug 22

Fare frontend - step1

⌚ Sep 1

Collegare con rabbit anche gestioneCredenziali con richieste di aiuto perchè deve aggiornargli i nuovi account creati

⌚ Sep 13

Finire frontend

⌚ Oct 1

Dockerizzare frontend

⌚ Oct 11

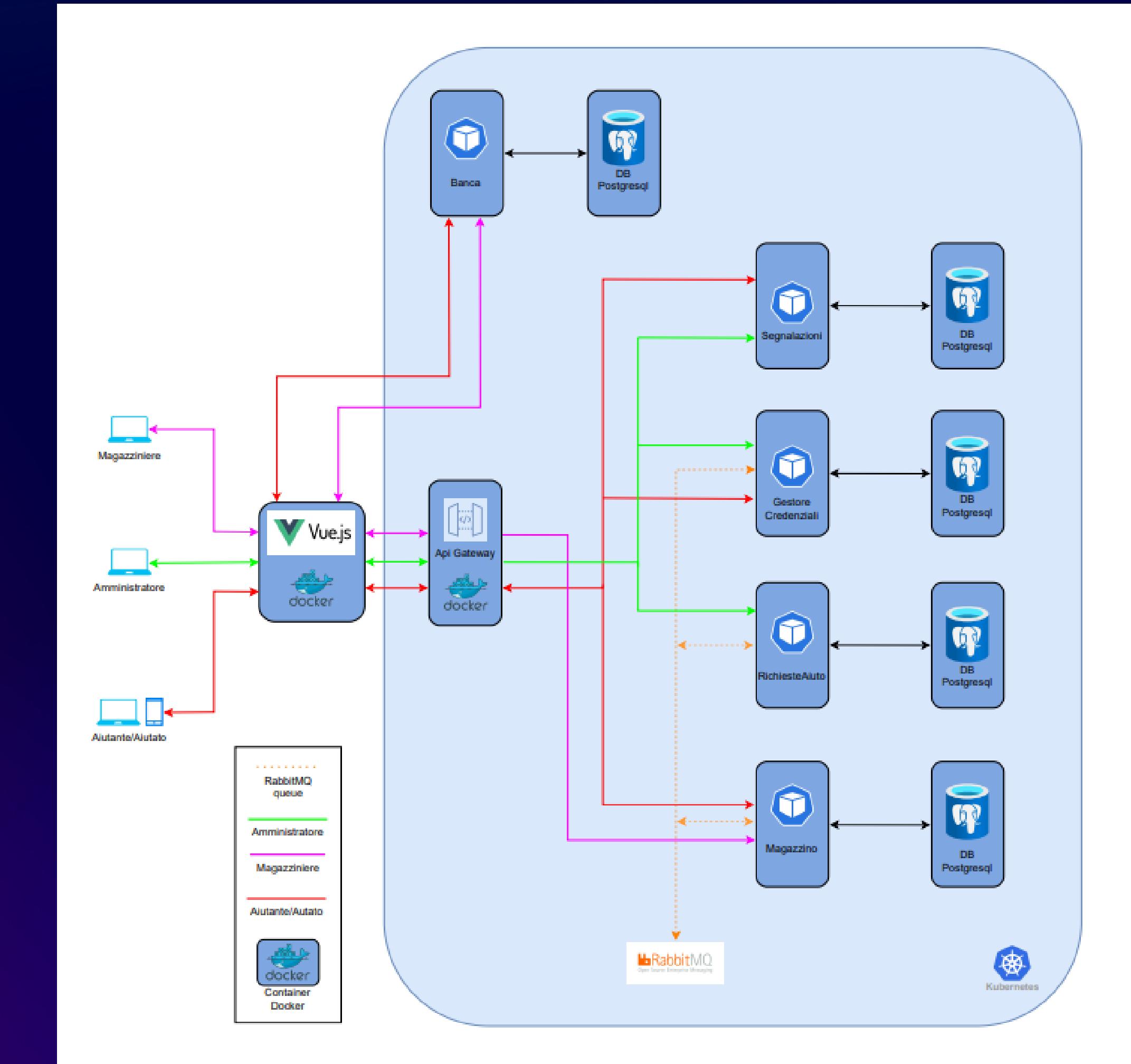
<https://trello.com/b/f7bumNWI/project-taas>

Indice

Backend

- 3 Architettura a microservizi
- 4 Microservizi
- 5 Strumenti di supporto:
 - Api Gateway
 - RabbitMQ
 - Eureka Server
- 6 Dataflow
- 7 Funzionalità aggiuntive:
 - File multimediali

Architettura a microservizi



Indice

Backend

- 3 Architettura a microservizi
- 4 Microservizi
- 5 Strumenti di supporto:
 - Api Gateway
 - RabbitMQ
 - Eureka Server
- 6 Dataflow
- 7 Funzionalità aggiuntive:
 - File multimediali

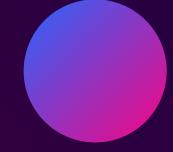
Microservizi



Banca



Gestione credenziali



Segnalazioni



Richieste di aiuto



Magazzini

Microservizi

Banca

Microservizio esterno che simula i conti in una banca.

Si occupa di:

- creare un nuovo conto corrente
- ricevere denaro e aggiungerlo a un conto (specificato) esistente
- restituire il saldo di un conto corrente richiesto

Nel progetto serve per poter effettuare donazioni da parte dell'utente
e ricevere denaro da parte del magazziniere.

Microservizi

Gestione credenziali

Microservizio che gestisce tutti gli utenti con le relative informazioni.

Si occupa di:

- creare un nuovo utente
- effettuare il login di un utente restituendo le informazioni a lui associate
- modificare le informazioni associate all'utente
- restituire tutti gli utenti con le relative informazioni
- aggiornare la categoria degli utenti
- modificare lo stato degli utenti (da approvare, approvato, in aggiornamento, bloccato)

Nel progetto serve per salvare le informazioni relative ad ogni utente.

Microservizi

Gestione credenziali

Quando un utente viene approvato viene inviato un messaggio con RabbitMQ al microservizio delle richieste di aiuto con alcune informazioni sull'utente per evitare di sovraccaricare il microservizio corrente.

- modificare lo stato degli utenti (da approvare, approvato, in aggiornamento, bloccato)

Microservizi

Microservizio che gestisce tutte le segnalazioni degli utenti.

Si occupa di:

- creare ed eliminare le segnalazioni degli utenti
- restituire una lista con tutte le segnalazioni

Segnalazioni

Esistono due tipi di segnalazioni:

- richiesta (sulla base della descrizione di questa)
- utente (a seguito di un servizio concluso)

Nel progetto serve per controllare la serietà degli utenti e delle richieste pubblicate.

Microservizi

Richieste di aiuto

Microservizio che gestisce le richieste di aiuto degli utenti.

Si occupa di:

- creare nuove richieste di aiuto e inviarle con RabbitMQ al microservizio del magazzino
- modificare lo stato di una richiesta (pubblicata, accettata, terminata)
- restituire le richieste di aiuto eventualmente già filtrate

Nel progetto serve per poter pubblicare e accettare richieste di aiuto.

Microservizi

Richieste di aiuto

Microservizio che gestisce le richieste di aiuto degli utenti.

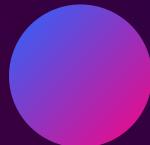
Si occupa di:

- creare nuove richieste di aiuto e inviarle con RabbitMQ al microservizio del magazzino

Richiesta di aiuto:

- (id)
- descrizione
- giorno
- indirizzo
- stato
- materiale (id)
- account accettante
- account pubblicante
- categoria

Microservizi



Magazzini

Microservizio che gestisce i magazzini con i relativi materiali.

Si occupa di:

- creare ed eliminare magazzini e materiali in essi contenuti
- restituire tutti i materiali disponibili dati giorno e provincia

Nel progetto serve per far inserire materiali sia all'utente in fase di creazione della richiesta di aiuto sia per l'aggiunta di nuovi materiali da parte del magazziniere.

Indice

Backend

- 3 Architettura a microservizi
- 4 Microservizi
- 5 Strumenti di supporto:
 - Api Gateway
 - RabbitMQ
 - Eureka Server
- 6 Dataflow
- 7 Funzionalità aggiuntive:
 - File multimediali

Strumenti di supporto



Api Gateway

Microservizio che svolge la funzione di singolo endpoint per le richieste API.

- Smista le richieste al microservizio opportuno
- Riceve la risposta dal microservizio
- Inoltre la risposta al chiamante

Implementa il proxy composition pattern.

Strumenti di supporto



RabbitMQ

Consente la comunicazione asincrona tra microservizi:

è p

- Implementa delle code dove vengono salvati i messaggi tra i vari microservizi
- monitoraggio dei vari processi e il loro utilizzo delle code di messaggi
- è possibile forzare la chiusura di connessioni e code di messaggi

Strumenti di supporto



Eureka Server

Noto come Discovey Server, tiene tutte le informazioni riguardo la comunicazione Client-Microservizi

- Strumento di supporto per l'API-Gateway
- Ogni micro servizio verrà registrato nel server Eureka e il server

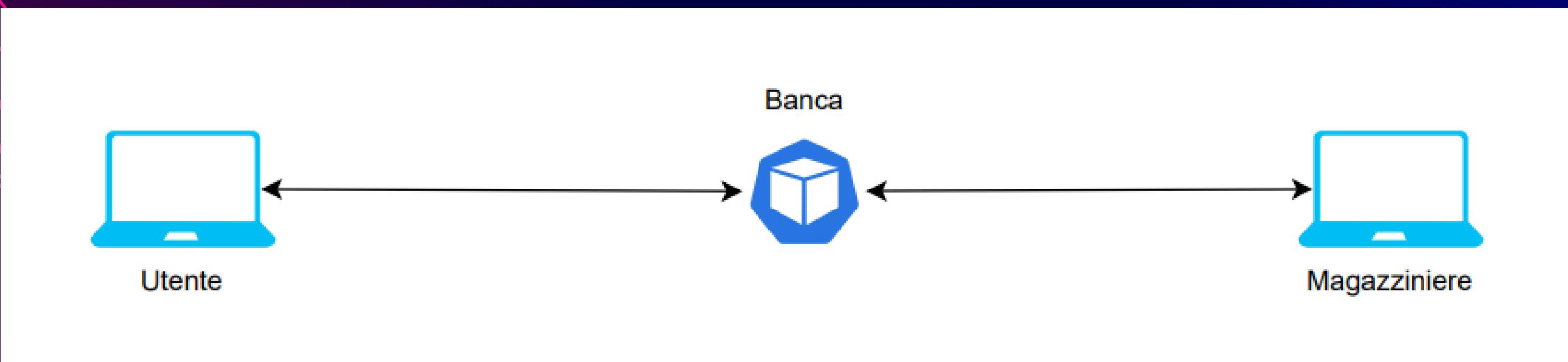
Eureka conosce tutte le applicazioni client in esecuzione su ciascuna porta e indirizzo IP.

Indice

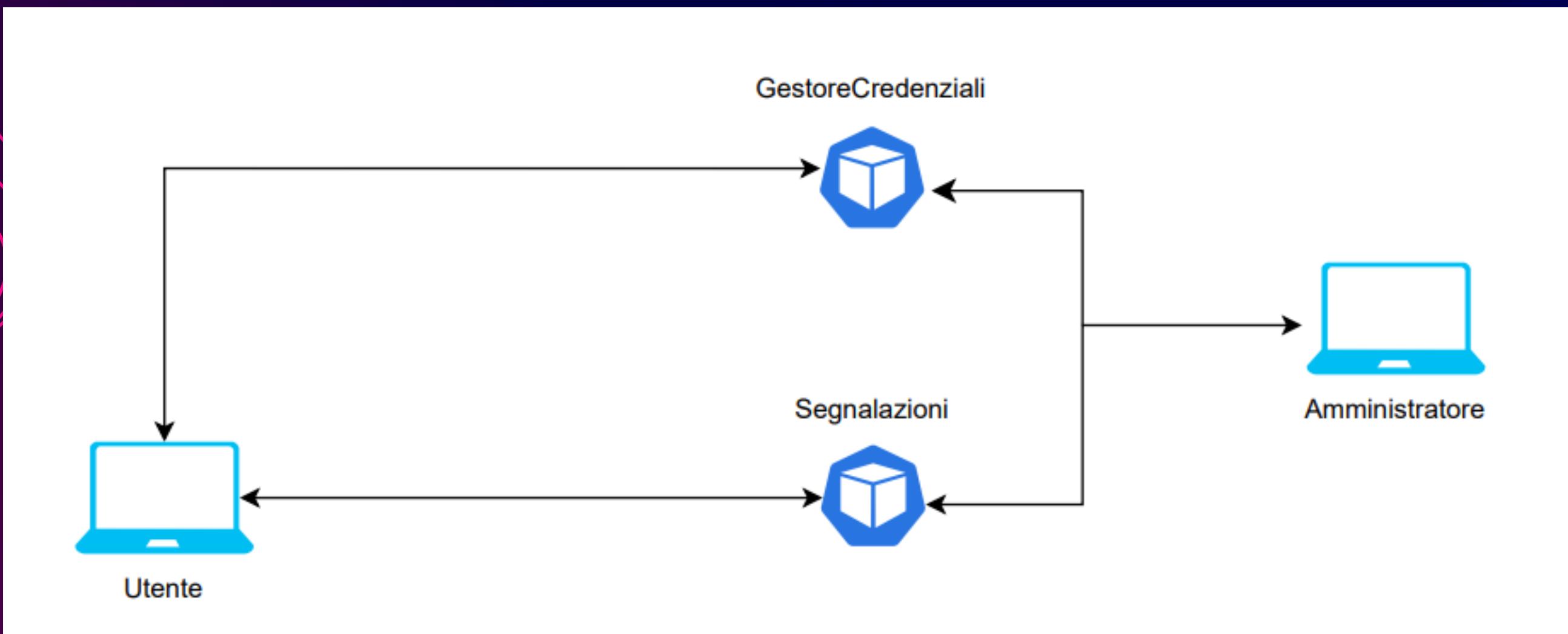
Backend

- 3 Architettura a microservizi
- 4 Microservizi
- 5 Strumenti di supporto:
 - Api Gateway
 - RabbitMQ
 - Eureka Server
- 6 Dataflow
- 7 Funzionalità aggiuntive:
 - File multimediali

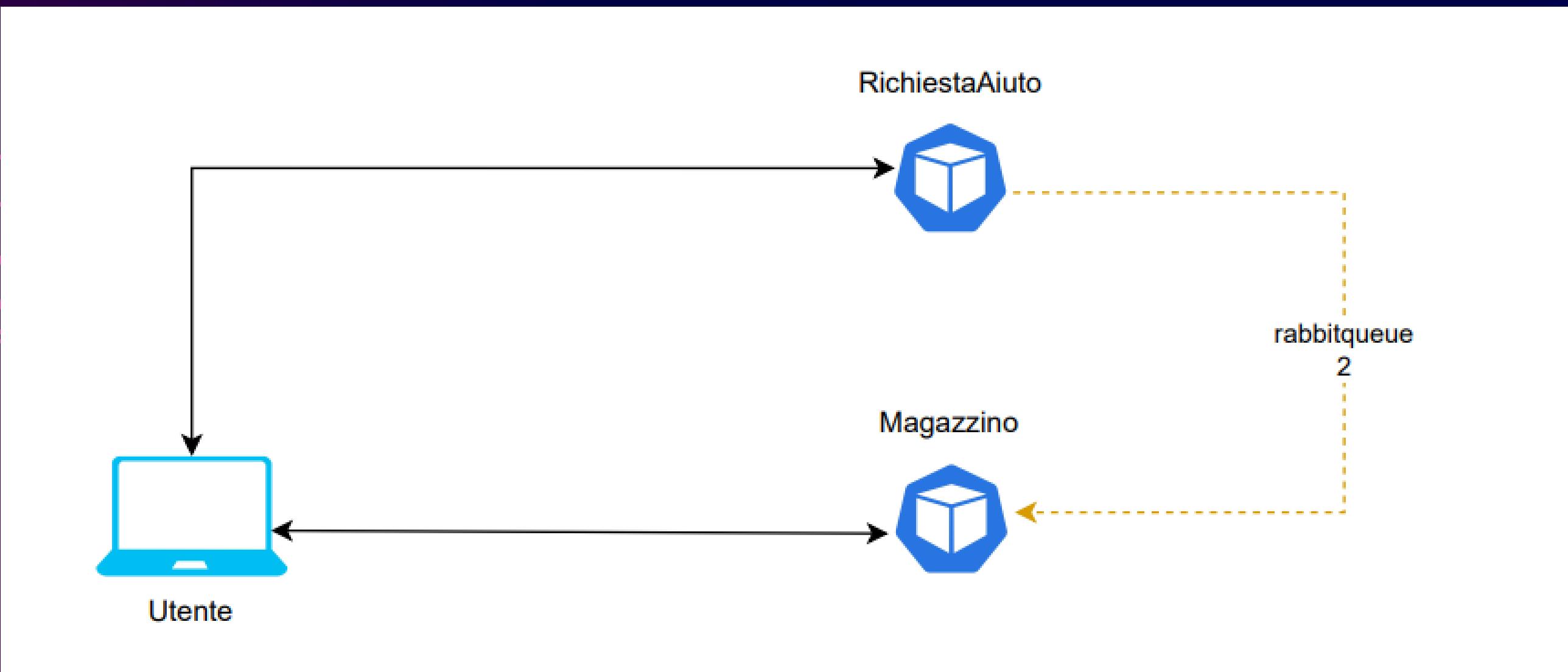
Dataflow: donazione



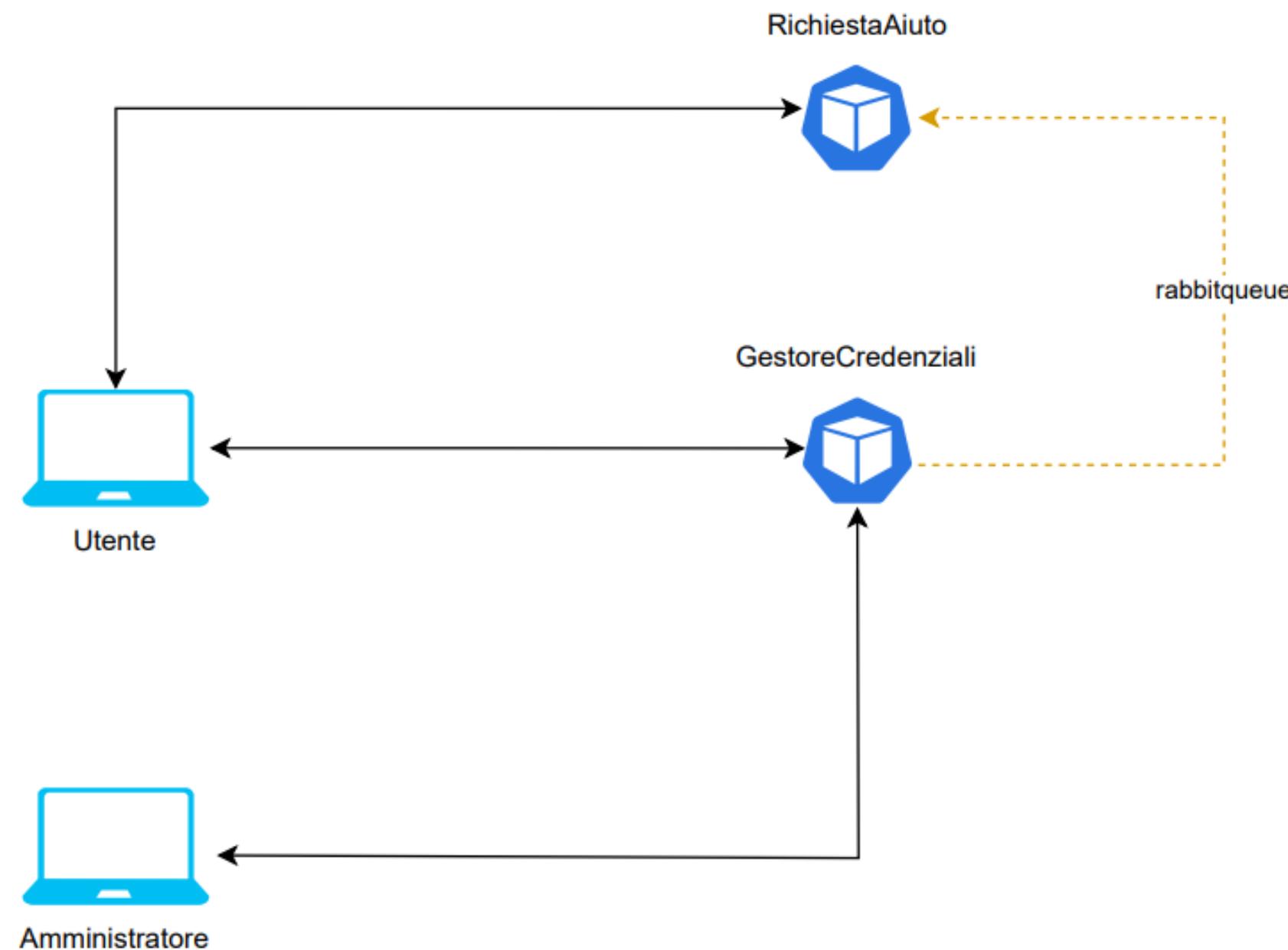
Dataflow: segnalazioni



Dataflow: richiesta d'aiuto



Dataflow: account



Indice

Backend

- 3 Architettura a microservizi
- 4 Microservizi
- 5 Strumenti di supporto:
 - Api Gateway
 - RabbitMQ
 - Eureka Server
- 6 Dataflow
- 7 Funzionalità aggiuntive:
 - File multimediali

Funzionalità aggiuntive: file multimediali

Abbiamo implementato le classi FileStorageService e FileController che utilizzano la libreria org.springframework.web.multipart.MultipartFile per salvare i curriculum e documenti di identità

```
public String storeFile(MultipartFile file, String type_doc, String name) {
    //Controlla che le cartelle Documents CV e Documenti_identita siano create altrimenti creaie
    checkDirectories();
    // Normalize file name
    String temp = "";
    if(!System.getProperty("user.dir").equals("\\\""))
        temp = System.getProperty("user.dir");
    String fileName = StringUtils.cleanPath(temp + "/Documents/" + type_doc + "/" + name + file.getOriginalFilename().substring(file.getOriginalFilename().lastIndexOf(".")));
    System.out.println(fileName);
    try {
        // Check if the file's name contains invalid characters
        if(fileName.contains("..")) {
            throw new FileStorageException("Sorry! Filename contains invalid path sequence " + fileName);
        }

        // Copy file to the target location (Replacing existing file with the same name)
        Path targetLocation = this.fileStorageLocation.resolve(fileName);
        Files.copy(file.getInputStream(), targetLocation, StandardCopyOption.REPLACE_EXISTING);

        return fileName;
    } catch (IOException ex) {
        throw new FileStorageException("Could not store file " + fileName + ". Please try again!", ex);
    }
}
```

```
@RestController
public class FileController {

    1 usage
    private static final Logger logger = LoggerFactory.getLogger(FileController.class);

    1 usage
    @Autowired
    private FileStorageService fileStorageService;

    # gianni
    @GetMapping("/downloadFile/{fileName:.+}")
    public ResponseEntity<Resource> downloadFile(@PathVariable String fileName, HttpServletRequest request) {
        // Load file as Resource
        Resource resource = fileStorageService.loadFileAsResource(fileName);

        // Try to determine file's content type
        String contentType = null;
        try {
            contentType = request.getServletContext().getMimeType(resource.getFile().getAbsolutePath());
        } catch (IOException ex) {
            logger.info("Could not determine file type.");
        }

        // Fallback to the default content type if type could not be determined
        if(contentType == null) {
            contentType = "application/octet-stream";
        }

        return ResponseEntity.ok()
            .contentType(MediaType.parseMediaType(contentType))
            .header(HttpHeaders.CONTENT_DISPOSITION, "attachment; filename=\"" + resource.getFilename() + "\"")
            .body(resource);
    }
}
```

Indice

Frontend

8

Vue

9

Login con Google

10

Funzionalità aggiuntive:

- Gestione della sessione utente
- Gestione del routing tra le pagine
- Gestione delle richieste scadute

11

Demo

Vue.js



Cos'è Vue

- Framework JavaScript per la creazione di interfacce utente.
- Standard HTML, CSS e JavaScript
- Modello di programmazione dichiarativo e basato su componenti

Perché Vue

- Responsive
- Single-Page Application
- Single-File Components
- Options API e Composition API come API style

Indice

Frontend

8

Vue

9

Login con Google

10

Funzionalità aggiuntive:

- Gestione della sessione utente
- Gestione del routing tra le pagine
- Gestione delle richieste scadute

11

Demo

Login con Google: OAuth2



- OAuth2 è un protocollo di autorizzazione che consente alle applicazioni di richiedere l'accesso a risorse protette da Google
- Nel progetto è stato usato per ottenere:
 - nome
 - cognome
 - emaildell'utente

Login con Google: OAuth2

Google Cloud taass ▾ Cerca (/) risorse, documenti, prodotti e altro Cerca

API API e servizi Credenziali + CREA CREDENZIALI - ELIMINA ↪ RIPRISTINA CREDENZIALI ELIMINATE

API e servizi abilitati Crea le credenziali per accedere alle API abilitate. [Ulteriori informazioni](#)

Libreria

Credenziali Chiavi API

<input type="checkbox"/> Nome	Data di creazione ↓	Limitazioni	Azioni
Nessuna chiave API da visualizzare			

Schermata consenso OAuth

Contratti sull'uso delle pagine

ID client OAuth 2.0

<input type="checkbox"/> Nome	Data di creazione ↓	Tipo	ID client	Azioni
Client taass	6 mag 2023	Applicazione web	944204629673-grd6...	

Account di servizio

<input type="checkbox"/> Email	Nome ↑	Azioni
Nessun account di servizio da visualizzare		

[Gestisci account di servizio](#)

Login con Google: vue3GoogleLogin



- Libreria di Vue utilizzata per semplificare l'autenticazione con Google
- Fornisce un'interfaccia per gestire il processo di accesso a Google, inclusa la configurazione del client ID

Login con Google: vue3GoogleLogin

```
googleTokenLogin()
  .then(async (response) => {
    const res = await fetch(
      "https://www.googleapis.com/oauth2/v3 userinfo?
      access_token=" + response.access_token, {
        method: "GET"
      }
    );
    const user = await res.json();
    this.email = user.email.toString();
    this.nome = user.given_name.toString();
    this.cognome = user.family_name.toString();
    ...
  });
}

export default {
```

- Processo di autenticazione con Google:
 - Reindirizzamento dell'utente a una pagina di accesso Google
 - Ottenimento di un token di accesso in risposta
 - Recupero delle informazioni dell'utente da Google utilizzando il token
- Tra le informazioni utente ottenute sono state utilizzate quelle relative a email, nome e cognome

Indice

Frontend

8

Vue

9

Login con Google

10

Funzionalità aggiuntive:

- Gestione della sessione utente
- Gestione del routing tra le pagine
- Gestione delle richieste scadute

11

Demo

Funzionalità aggiuntive: sessione utente



- Libreria per la gestione dello stato in Vue.js
- Fornisce uno store centrale per archiviare dati condivisi e modificarli tramite mutazioni. Gestisce quindi uno stato globale dell'applicazione che è accessibile e modificabile da qualsiasi componente in modo reattivo.
- Semplifica la gestione dello stato condiviso all'interno dell'applicazione Vue.

Funzionalità aggiuntive: sessione utente

```
const store = new Vuex.Store({  
  state: {  
    userId: localStorage.getItem('userId') || null,  
    lastLoginTime: localStorage.getItem('lastLoginTime') || null,  
    magLoggedIn: localStorage.getItem('magLoggedIn') || false,  
    adminLoggedIn: localStorage.getItem('adminLoggedIn') || false  
  },  
  ...  
})
```

Inizializza un'istanza di store Vuex con uno stato iniziale che contiene informazioni su: userId, lastLoginTime, magLoggedIn e adminLoggedIn

Variabile che serve per gestire il logout automatico dopo 30 minuti di sessione utente aperta (per sicurezza)

Funzionalità aggiuntive: routing tra le pagine

- Navigazione tra le pagine
- Separazione delle responsabilità
- Controlli di accesso e autorizzazione

```
{  
  path: '/magazzino',  
  name: 'magazzino',  
  component: MagazzinoView,  
  beforeEnter(to, from, next) {  
    if (app.$store.state.magLoggedIn === 'true')  
      next({name: 'magazzino-home'});  
    else  
      next();  
  }  
}  
  
{  
  path: '/magazzino/magazzino-home',  
  name: 'magazzino-home',  
  component: MagazzinoHomeView,  
  beforeEnter(to, from, next) {  
    if (app.$store.state.magLoggedIn === false)  
      next({name: 'magazzino'});  
    else  
      next();  
  }  
}
```

Funzionalità aggiuntive: richieste scadute

Per evitare di sovraccaricare il backend, quando si prendono tutte le richieste da frontend si controlla se tra quelle non terminate ci sono nuove richieste “scadute” che quindi vanno fatte terminare.

```
let today = new Date().getFullYear().toString() + '-' + (new Date().getMonth() + 1).toString() + '-' + new  
Date().getDate().toString();  
  
if ((listaRichiesteTotali[i].stato !== 'terminata') && (listaRichiesteTotali[i].giorno !== today) && (new  
Date(listaRichiesteTotali[i].giorno) < new Date())) {  
    listaRichiesteTotali[i].stato = 'terminata';  
    this.terminaRichiesta(listaRichiesteTotali[i].id);  
}
```

Indice

Frontend

8

Vue

9

Login con Google

10

Funzionalità aggiuntive:

- Gestione della sessione utente
- Gestione del routing tra le pagine
- Gestione delle richieste scadute

11

Demo

A large, abstract graphic on the left side of the slide features a series of thin, curved lines that curve upwards and outwards from the bottom left corner. The lines are colored with a gradient, transitioning from dark purple at the bottom to bright magenta at the top. They create a sense of depth and motion, resembling a stylized ribbon or a series of overlapping planes.

Demo

Indice

Conclusione

12

Docker

13

Kubernetes

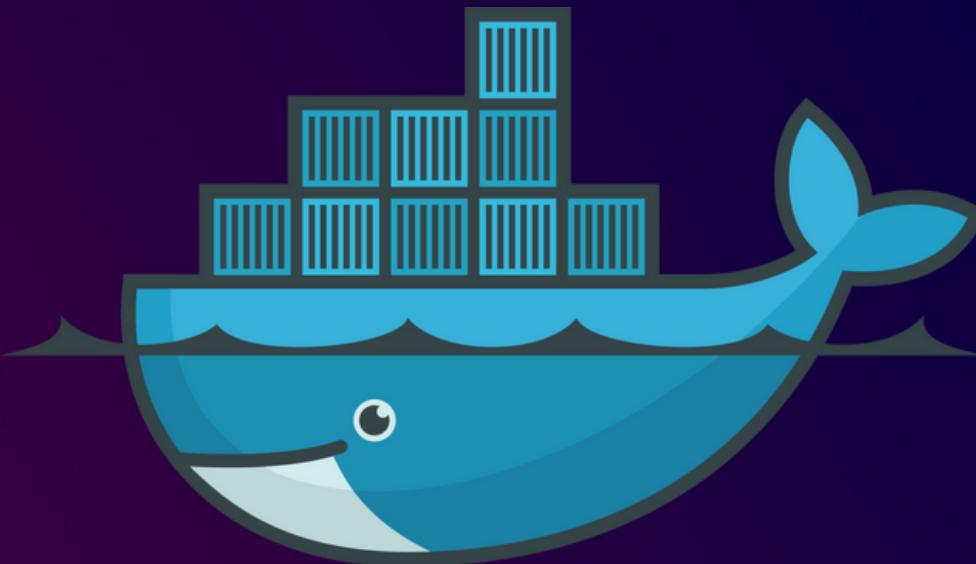
14

GitHub e README

15

Sviluppi futuri

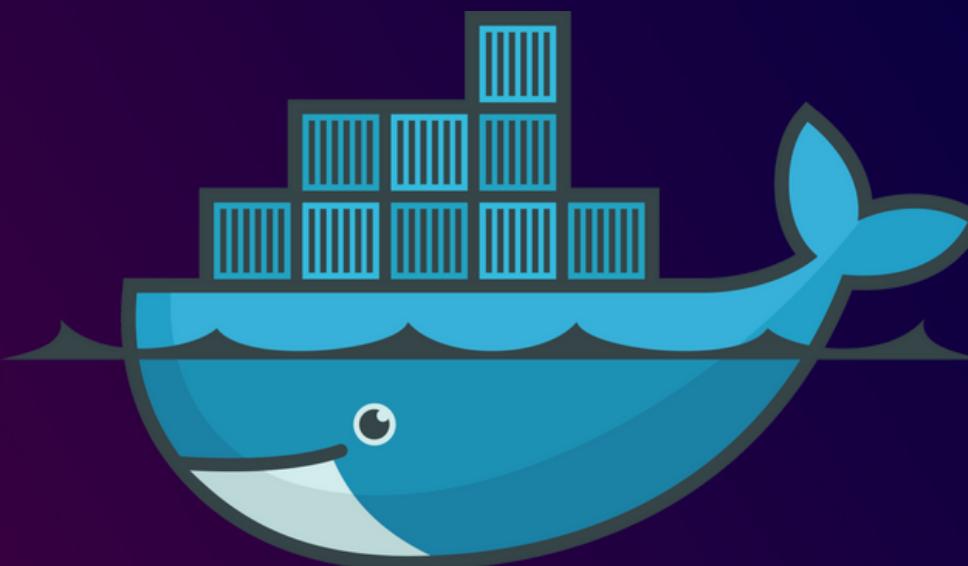
Docker



docker

- Utilizzato per la creazione delle immagini testare e implementare l'applicazione
- Utilizzato per debugging e sviluppo tramite Docker Repository
 - <https://hub.docker.com/repository/docker/giargiapower/helpmeandyou/general>

Docker-compose



docker

- Builda le immagini dei microservizi
- Espone tutti i microservizi su porte 808* ed i rispettivi database postgres
- Api gateway esposto sulla porta 30000
- Frontend vue esposto sulla porta 8080
- Ottiene le immagini da Dockerhub
- Avvia tutti i container e le relative reti

```
app:
  build: .
  image: magazzino
  container_name: magazzino
  ports:
    - "8087:8087"
  depends_on:
    - dbpostgresql
  networks:
    - magazzino
  environment:
    - SPRING_DATASOURCE_URL=jdbc:postgresql://dbpostgresql:5432/postgres
    - SPRING_DATASOURCE_USERNAME=postgres
    - SPRING_DATASOURCE_PASSWORD=postgres
    - SPRING_JPA_HIBERNATE_DDL_AUTO=update
  volumes:
    db-data:
    pgadmin-data:
  networks:
    magazzino:
      driver: bridge
```

```
services:
  dbpostgresql:
    container_name: postgres_mag
    image: "postgres"
    ports:
      - "5432"
    volumes:
      - db-data:/var/lib/postgresql/data
    networks:
      - magazzino
    environment:
      POSTGRES_DB: postgres
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    restart: unless-stopped
```

- Ogni microservizio ha:
 - una sua network interna (in questo caso magazzino)
 - un proprio DB postgres ad eccezione dell'apigateway

Dockerfile backend

```
FROM openjdk:17-alpine
RUN addgroup -S spring && adduser -S spring -G spring
VOLUME /tmp
EXPOSE 8080
ARG DEPENDENCY=target
ADD ${DEPENDENCY}/*.jar magazzino.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/magazzino.jar"]
```

- Aggiunge un gruppo e un utente spring (per comodità).
- la directory /tmp sarà il volume dell'immagine Docker. I volumi sono usati per conservare dati che possono essere condivisi tra il container e l'host o tra container
- avvia il microservizio con il JAR specificato

Dockerfile frontend

```
FROM node:16

# Set the working directory in the container
WORKDIR /frontend-vue

# Copy package.json and package-lock.json to the working directory
COPY package*.json .

RUN npm cache clean --force
# Install app dependencies
RUN npm install

# Copy the rest of the application code
COPY .

# Expose the port the app runs on
EXPOSE 8080

# Define the command to run your app
CMD ["npm", "run", "serve"]
```

- Imposta la directory di lavoro all'interno del container su frontend-vue
- Pulisce la cache per ridurre la dimensione dell'immagine Docker e rimuovere eventuali cache obsolete
- Definisce il comando predefinito da eseguire quando il container viene avviato. In questo caso, avvia l'applicazione con il comando npm run serve è

Indice

Conclusione

12

Docker

13

Kubernetes

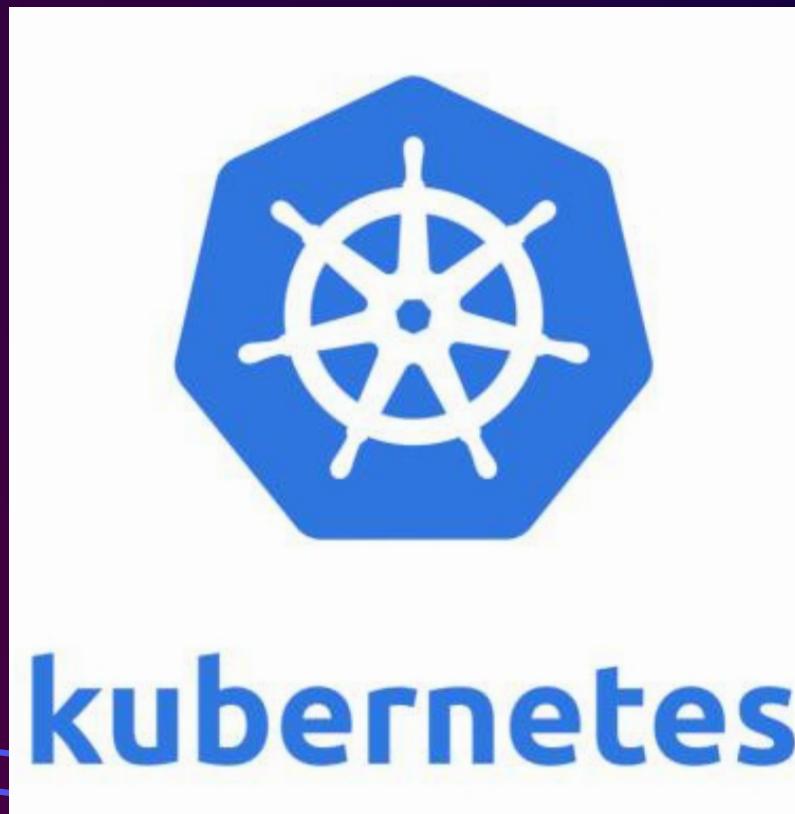
14

GitHub e README

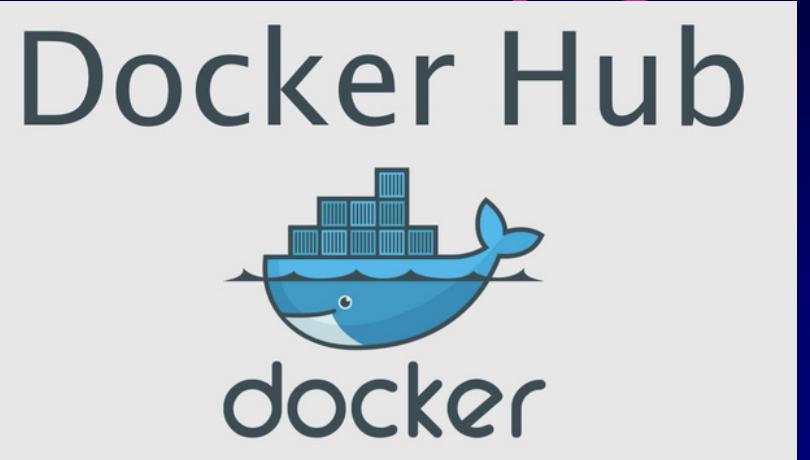
15

Sviluppi futuri

Kubernetes



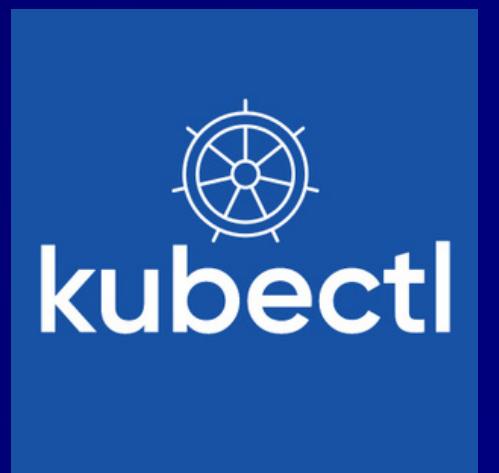
Docker hub utilizzato come istanza locale di Kubernetes



Kompose è stato usato per la conversione Docker-Kubernetes



kubectl è una interfaccia CLI per eseguire e interagire con Kubernetes

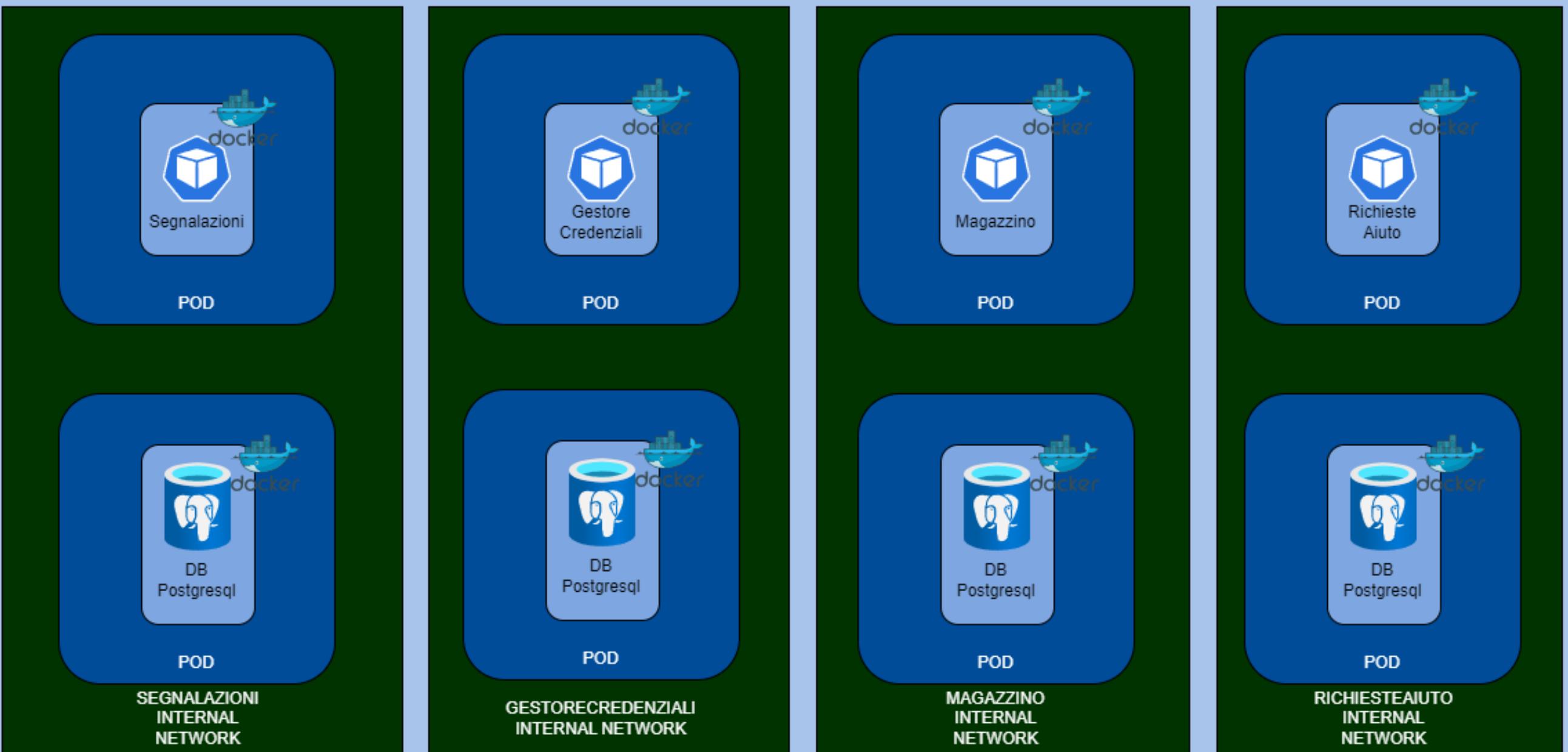


Kompose

- Genera 4 componenti fondamentali :
 - **Deployment**: Uno per ogni microservizio, database e RabbitMQ che descrive lo stato desiderato e mantenuto dal controller
 - **Service** : consente l'accesso e il routing ai pods relativi al deployment
 - **PersistentVolumeClaim**: Mappa su kubernetes i volumi utilizzati su docker in particolare per il salvataggio dei dati del database
 - **NetworkPolicy**: Mappa su kubernetes le reti definite su docker tramite il docker-compose



KUBERNETES NODES



Kubernetes

Avvio dell'app

- Lanciare Docker Desktop e attivare il nodo di Kubernetes integrato
- Lanciare `docker pull --all-tags giargiapower/helpmeandyou` per scaricare tutte le immagini necessarie per il funzionamento dell'applicazione
- Lanciare `kubectl apply -f ./deployment.yaml` per far partire i pod di Kubernetes
- Entrare nella cartella demo e lanciare `kubectl apply -f ./kubernetes-manifests.yaml` per far partire la demo simulativa della banca
- Per far partire il frontend entrare nell'cartella frontend-vue e lanciare rispettivamente :
 - `docker-compose build`
 - `docker-compose up`

Indice

Conclusione

12

Docker

13

Kubernetes

14

GitHub e README

15

Sviluppi futuri

GitLab e README

- I file dell'applicazione sono tutti su:
<https://gitlab2.educ.di.unito.it/sp223979/helpmeandyou.git>
- Ogni microservizio ha:
 - una cartella personale (con sottocartelle)
 - README con tutte le informazioni del microservizio e i test da fare con Postman per il backend
- Backend e Frontend sono stati divisi in cartelle separate così da poter lavorare in parallelo

Indice

Conclusione

12

Docker

13

Kubernetes

14

GitHub e README

15

Sviluppi futuri

Sviluppi Futuri

- Aggiungere altri metodi di pagamento (Satispay, Google Pay, PayPal etc...)
- Mappa con OpenStreetMaps
- Notifiche in tempo reale per gli utenti
- Sistema di scambio email/chat tra gli utenti per mettersi d'accordo su eventuali modifiche o ritardi



Grazie per l'attenzione



HelpMe&You

