OST
Ostschweizer
Fachhochschule

TaskTracker

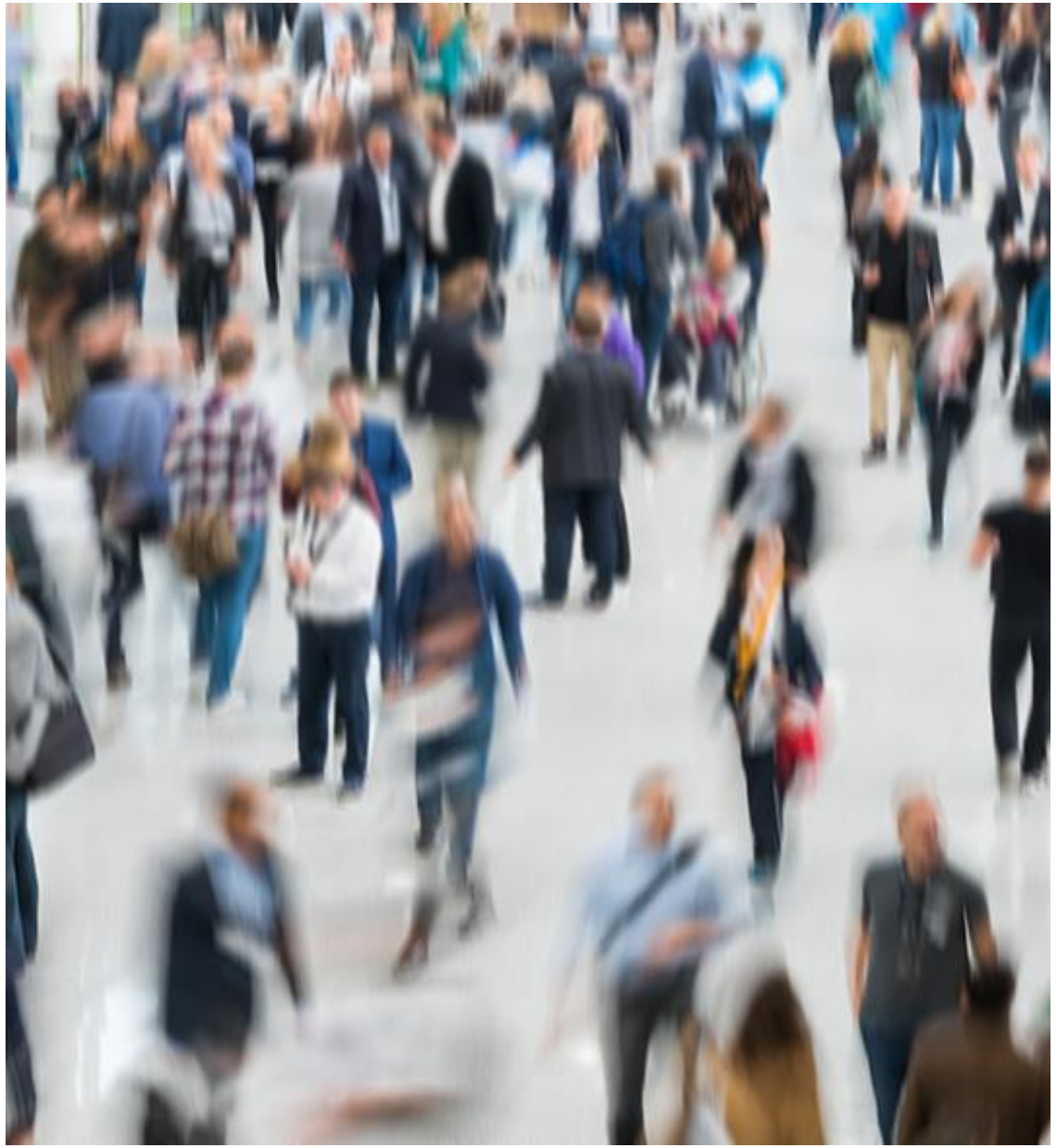
Distributed Systems – Challenge Task

13.05.2024

Stefan Meyer, Nicolas Richard, Dominik Rüegg

Inhalt

1. Anforderungen
2. Projekt Übersicht
3. Infrastruktur
4. Architektur
5. Scheduled Task
6. Live Demo

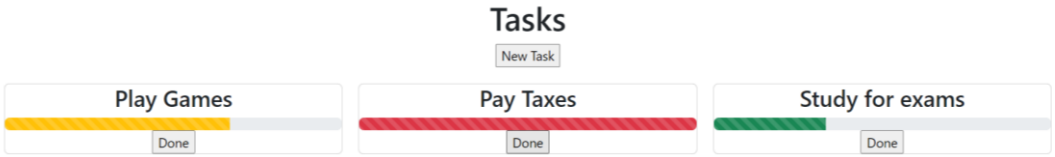


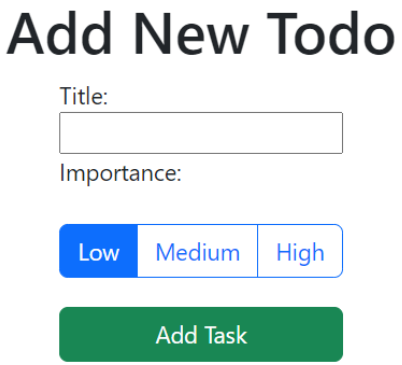
1 Anforderungen

- Loadbalancing mit mehreren Instanzen und Failover einer Service-Instanz
- Dockerisiert
- Einfaches Frontend
- Scheduled Task
- Persistente Speicherung
- Verwendung der neuesten stabilen Versionen der ausgewählten Bibliotheken und Frameworks

2 Projekt Übersicht

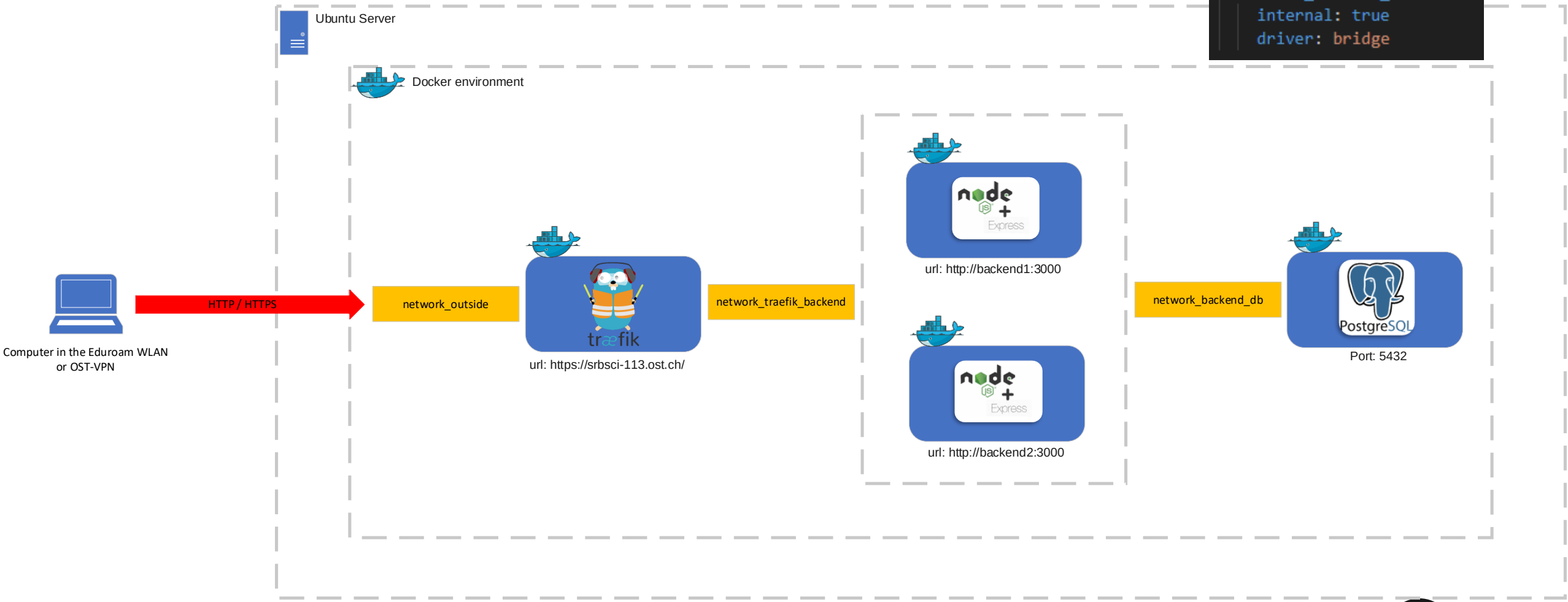
- Taskapp
- Erfassung von Tasks
- Sobald 5 oder mehr Tasks mit “high” Priorität wird ein dummy Mail versendet
- Persönliche Ziele
  - Neue Technologien ausprobieren
  - Erweiterung der bestehenden Kenntnisse



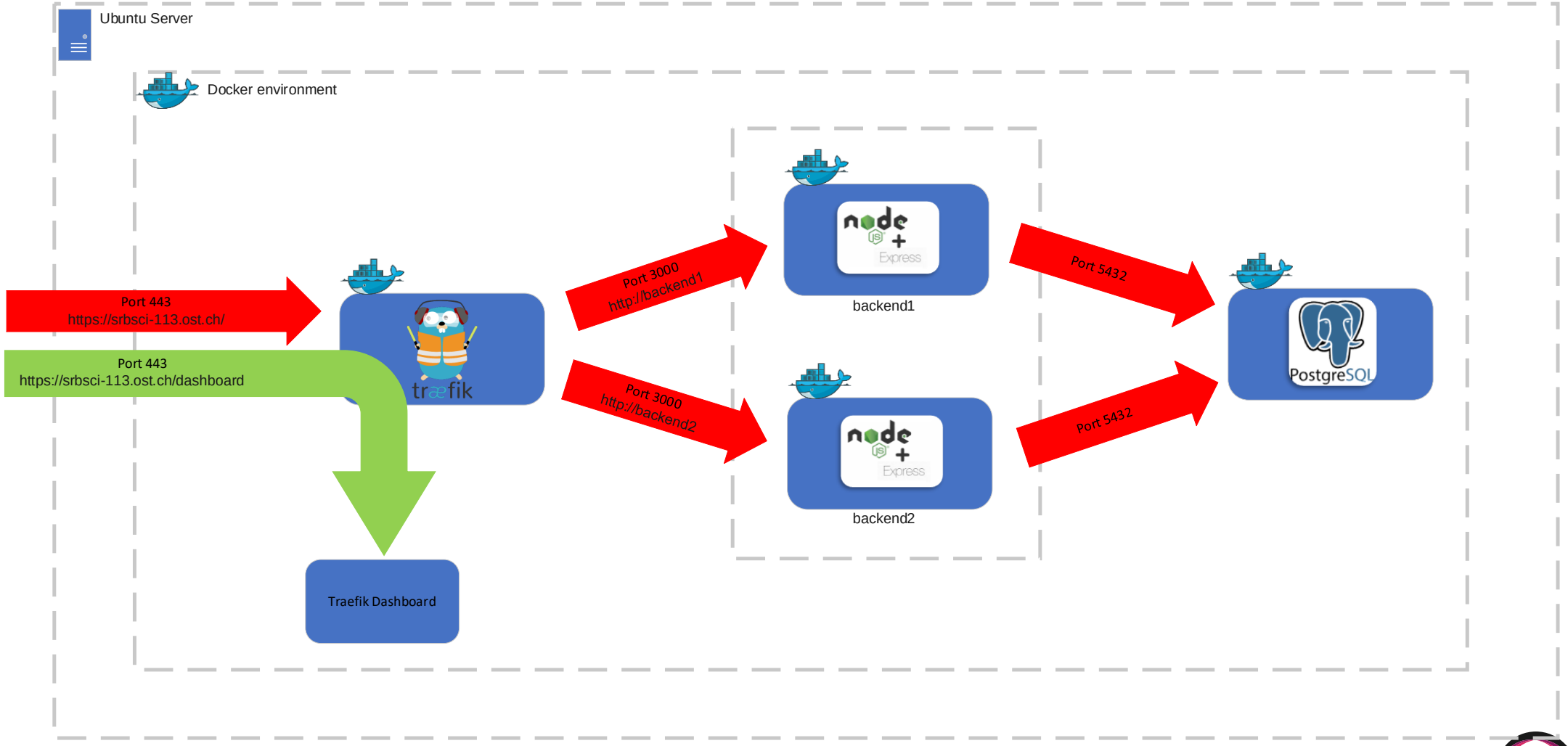


3 Infrastruktur - Übersicht

```yaml
networks:
  network_outside:
    internal: false
    driver: bridge
  network_traefik_backend:
    internal: false
    driver: bridge
  network_backend_db:
    internal: true
    driver: bridge
```



3 Infrastruktur – Traffic flow



3 Infrastruktur – Traefik

```yaml
traefik:
  image: traefik:v2.11
  restart: always
  networks:
    - network_outside
    - network_traefik_backend
  ports:
    - "80:80"
    - "443:443"
  volumes:
    - "/var/run/docker.sock:/var/run/docker.sock:ro"
    - "./traefik.yaml:/etc/traefik/traefik.yaml"
    - "./traefik_dynamic.yaml:/etc/traefik/traefik_dynamic.yaml"
  deploy:
    resources:
      limits:
        cpus: '0.5'
        memory: '512M'
```

3 Infrastruktur – Traefik

traefik.yaml

```yaml
experimental:
  http3: true

entryPoints:
  web:
    address: ":80"
    http:
      redirections:
        entryPoint:
          to: websecure
          scheme: https
  websecure:
    address: ":443"
    http3:
      advertisedPort: "443"
api:
  dashboard: true

providers:
  file:
    filename: "/etc/traefik/traefik_dynamic.yaml"

log:
  level: TRACE
```

3 Infrastruktur – Traefik

traefik_dynamic.yaml

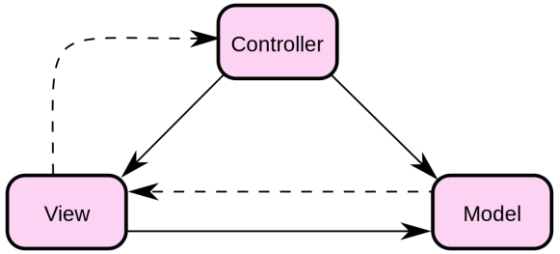
```yaml
http:
  routers:
    dashboard:
      rule: "Host(`srbsci-113.ost.ch`) && PathPrefix(`/api`, `/dashboard`)"
      service: "api@internal"
      middlewares:
        - auth
      tls:
        domains:
          - main: "srbsci-113.ost.ch"
    service_taskapp:
      rule: "Host(`srbsci-113.ost.ch`)"
      service: service_taskapp
      entrypoints:
        - "websecure"
      tls:
        domains:
          - main: "srbsci-113.ost.ch"
  services:
    service_taskapp:
      loadBalancer:
        healthcheck:
          path: "/"
          interval: "3s"
          timeout: "1s"
        servers:
          - url: "http://backend1:3000"
          - url: "http://backend2:3000"
  middlewares:
    auth:
      basicAuth:
        users:
          - "test:$2a$10$AnIXMhNXDQsOAgBggMLOA.SwAiJlIGb0wztEqOpSdTXXxD5TBlVku"
```

3 Infrastruktur – Weiteres

- Build Pipeline
- Persistenter Storage
- Docker Secrets

4 Architektur

- Backend
- Nodejs Framework Express
- MVC Pattern
  - Handlebars




```
<form class="text-center" action="/delete/{{this.id}}"><input type="submit" value="Done"></form>
```

⇩

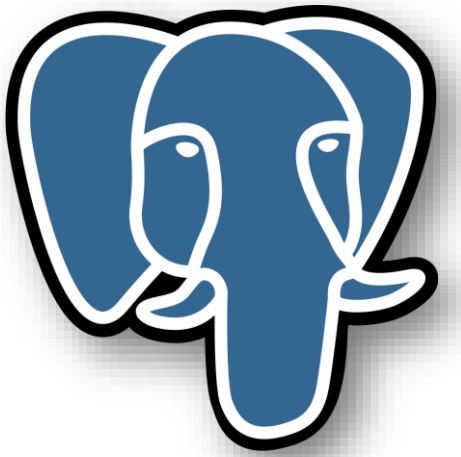
```
router.get('/delete/:id', indexController.delete);
```

⇩

```
async delete(req: Request, res: Response){
  await pool.query('DELETE FROM tasks WHERE id=$1', [req.params.id])
  res.redirect('/');
}
```

4 Architektur

- Database
- Postgresql
  - CreateDB.sql



CreateDB.sql

```sql
CREATE TABLE tasks (
    id SERIAL PRIMARY KEY,
    title VARCHAR(100),
    priority VARCHAR(100)
);

CREATE TABLE time (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO time (timestamp) VALUES (CURRENT_TIMESTAMP);
```

5 Scheduled Task

Klasse Email:

transporter: Nodemailer-Transporter
msg: E-Mail-Nachricht

Konstruktor():

Initialisiere den Nodemailer-Transporter
Konfiguriere die E-Mail-Nachricht

sendMail():

Versuche, eine E-Mail zu senden

```
export class email {
    transporter: any
    msg: any
    constructor() {
        this.transporter = nodemailer.createTransport({
                                host: "smtp.ethereal.email",
                                port: 587,
                                //Authentication});
        this.msg = {from: '"My Task App" <TaskApp@ethereal.email>',
                            to: "nicolas.richard@ost.ch",
                            subject: "Important Tasks",
                            text: "You have more than 5 important Tasks",
                        }
                }

    public async sendMail(): Promise<void> {
        try {
            const info = await this.transporter.sendMail(this.msg);
        } catch (error) {
            console.error("Error sending mail: ", error);
        }
    }

}
```

5 Scheduled Task

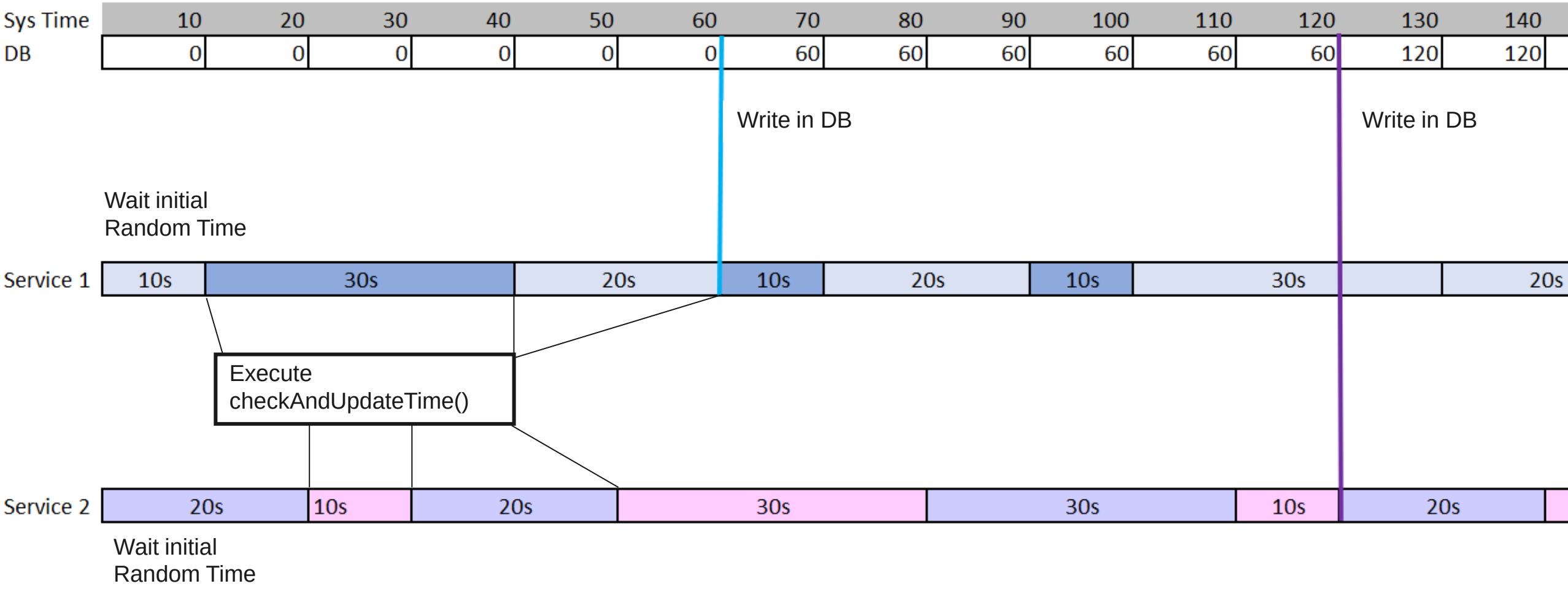
Klasse TaskScheduledjobs:
randomTime: Zufällige Zeit für die erste Überprüfung

Konstruktor():
Initialisiere Variablen und Einstellungen
Definiere Funktion für die periodische Überprüfung der Zeit
Definiere Funktion für das Senden von E-Mails

checkAndUpdateTime():
if time in DB > 60:
Aktualisiere die Zeit in der Datenbank
if „high“ priority Tasks > 5:
sendEmail
return timeout(checkAndUpdateTime(), **new** randomTime)

sendEmailJob():
Überprüfe jede Sekunde, ob eine E-Mail versendet werden muss

5 Scheduled Task



Live Demo