Time Tracking

Musa Rochi

Marcel Schubert

Nicola Aebi

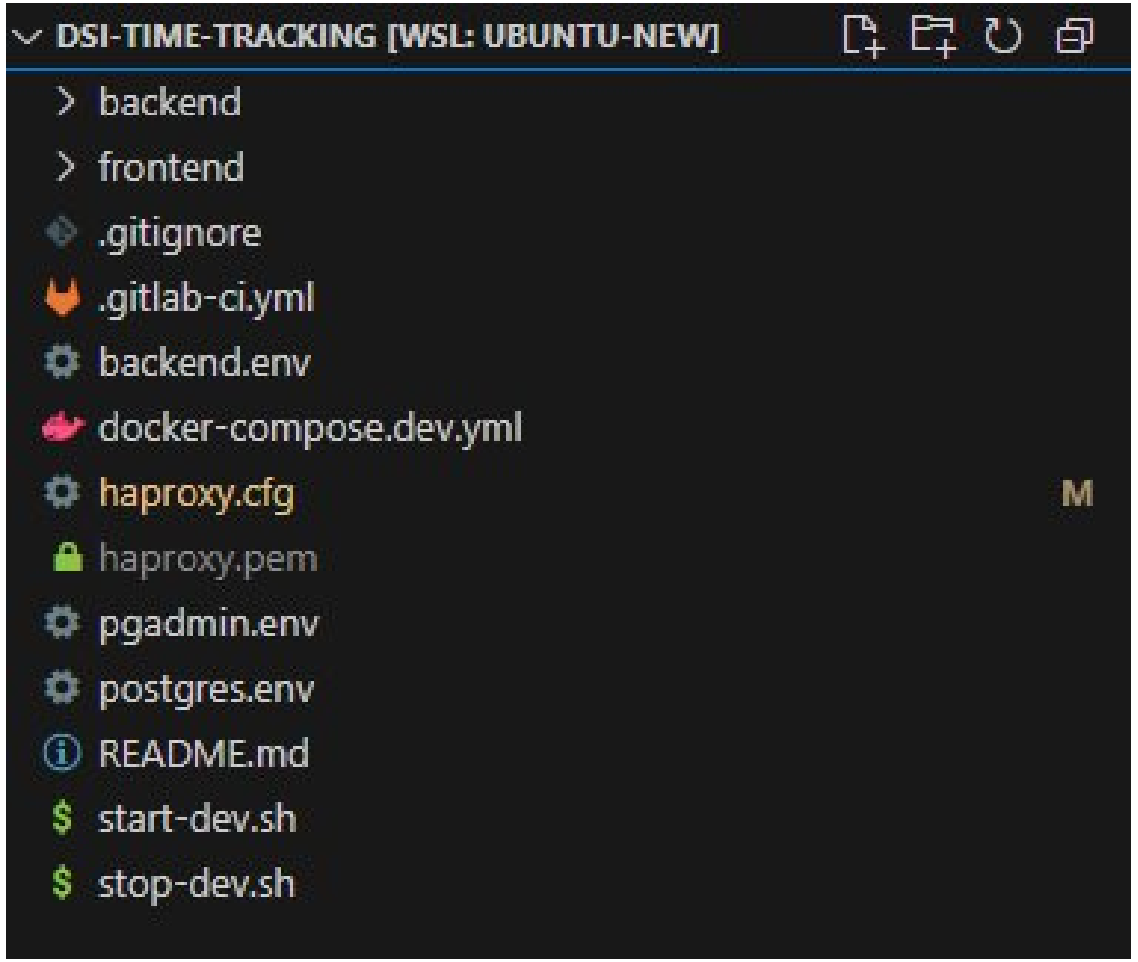
Inhalt

- Architecture
  - Tech-Stack
  - C4: System Context
  - C4: Container
  - DNS Service Discovery
  - Deployment
- Repository
- Database
- Backend
- Frontend
- Demo

Tech-Stack

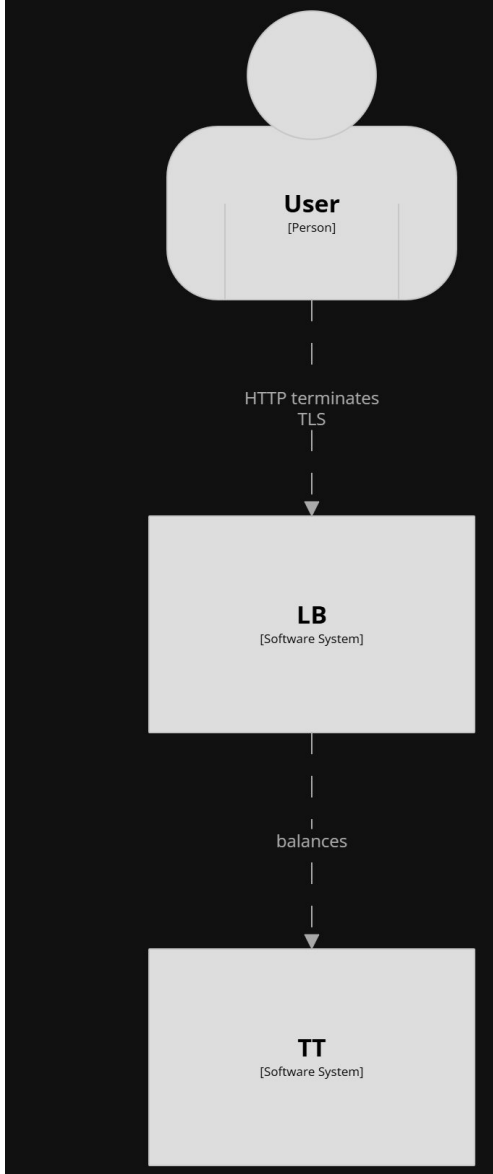
Repository

- Mono Repo
  - Einfache Handhabung
    - Docker Compose



C4: System Context

- User
  - Uses the service
  - Interacts with the backend via React SPA
- LB
  - Balances load to services
- TT
  - System provides SPA and backend API



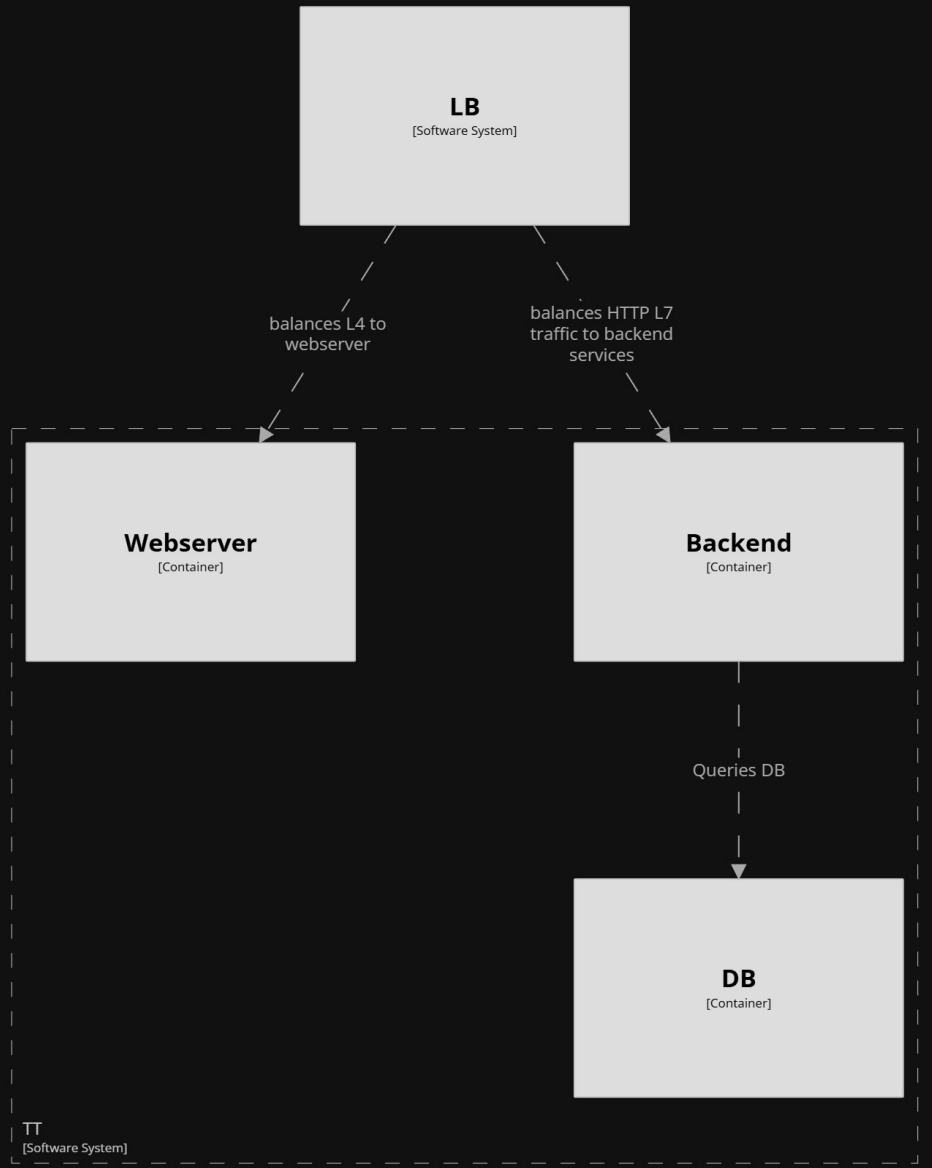
[System Landscape]
Freitag, 23. Mai 2025 um 18:47 Mitteleuropäische Sommerzeit

C4: Container

- Loadbalancer
  - HaProxy supports DNS service discovery
  - L7 load balancing to the backend using round robin
  - L4 load balancing to nginx webserver

- Backend
  - NodeJs, Typescript, Express
  - Stateless
  - Scaled by using docker compose

- Webserver
  - Nginx
  - Serves SPA



[Container] TT

Freitag, 23. Mai 2025 um 19:26 Mitteleuropäische Sommerzeit

DNS Service Discovery

- populate server addresses by querying DNS (internal Docker DNS)
- Docker: --scale flag is not documented well

```
docker compose -f docker-compose.dev.yml up
       --scale dsi-backend=2 -d
```

```
/app # nslookup dsi-backend
Server:         127.0.0.11
Address:        127.0.0.11:53

Non-authoritative answer:

Non-authoritative answer:
Name:   dsi-backend
Address: 172.18.0.6
Name:   dsi-backend
Address: 172.18.0.5
```

```
resolvers docker
    nameserver dns 127.0.0.11:53
    resolve_retries       3
    timeout resolve       1s
    timeout retry         1s
    hold valid            10s
backend http_b_backend_testing
    mode http
    balance roundrobin
    option httpchk GET /
    server-template dsi-backend- 2 dsi-backend:8080 check resolvers docker resolve-prefer ipv4
```

HAProxy version 3.1.7-c3f4089, released 2025/04/17

Statistics Report for pid 8

> General process information

pid = 8 (process #1, nbproc = 1, nbthread = 16)
uptime = 0d 0h16m08s; warnings = 2
system limits: memmax = unlimited; ulimit-n = 475
maxsock = 475; **maxconn** = 200; **reached** = 0; **maxpipes** = 0
current conns = 8; current pipes = 0/0; conn rate = 0/sec; bit rate = 0.815 kbps
Running tasks: 0/67 (0 niced); idle = 100 %

- active UP
- active UP, going down
- active DOWN, going up
- active or backup DOWN
- active or backup DOWN for maintenance (MAINT)
- active or backup SOFT STOPPED for maintenance
- backup UP
- backup UP, going down
- backup DOWN, going up
- not checked

Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:

- Scope :
- Hide 'DOWN' servers
- Disable refresh
- Refresh now
- CSV export
- JSON export (schema)

External resources:

- Primary site
- Updates (v3.1)
- Online manual

http_front

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	4	-	4	4	200	4			6 827	177 593	0	0	0					OPEN								

http_f_backend_testing

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	2	-	3	3	200	3			1 114	1 907	0	0	0					OPEN								

http_b_backend_testing

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
dsi-backend-1	0	0	-	0	1		2	2	-	2	2	3s	0	0		0		0	0	0	0	16m8s UP	L7OK/200 in 0ms	1/1	Y	-	0	0	0s	-
dsi-backend-2	0	0	-	0	2		0	1	-	2	2	3s	1 114	1 907		0		0	0	0	0	16m8s UP	L7OK/200 in 0ms	1/1	Y	-	0	0	0s	-
Backend	0	0		0	3		2	3	20	4	4	3s	1 114	1 907	0	0		0	0	0	0	16m8s UP		2/2	2	0		0	0s	

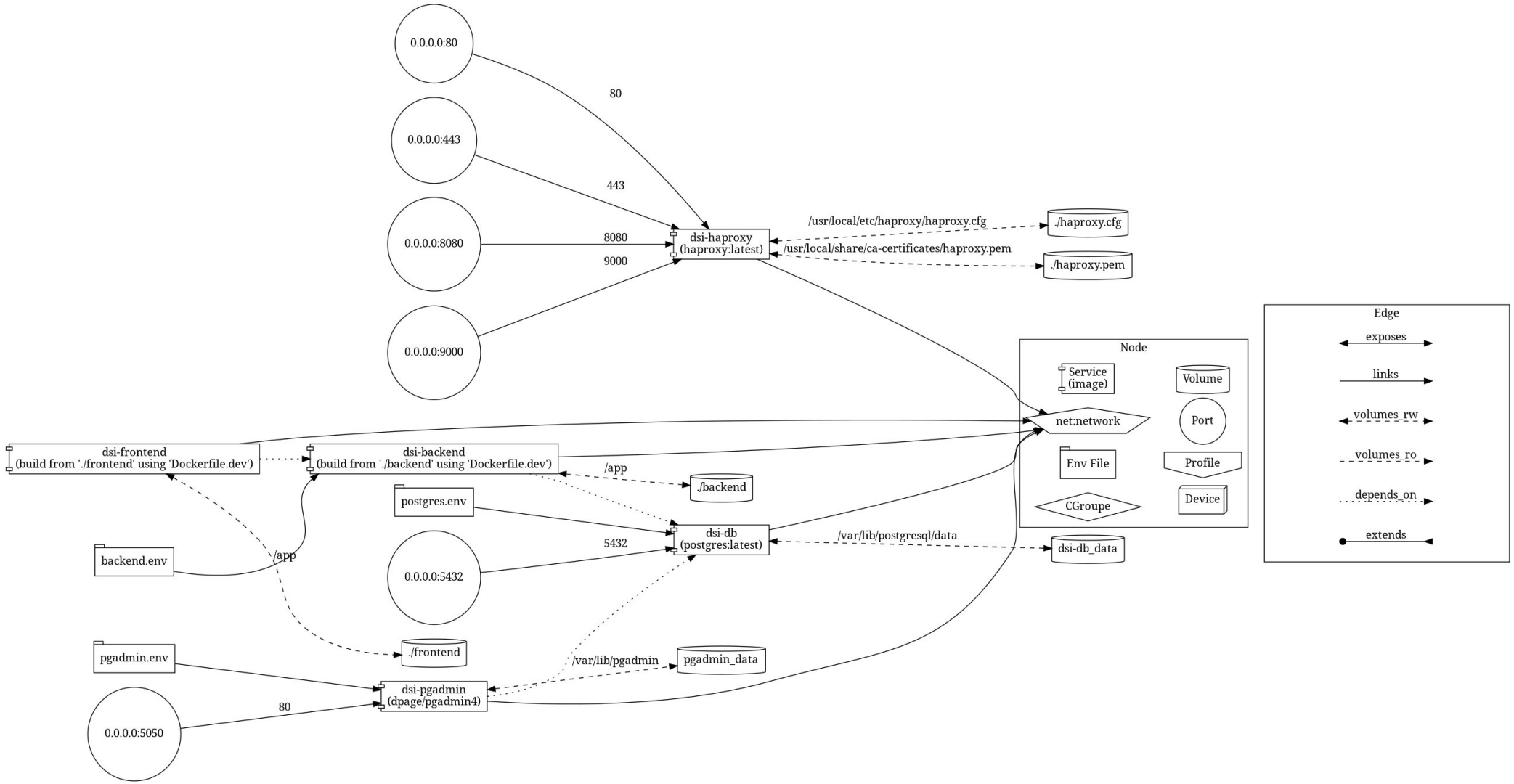
http_back

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
dsi-frontend	0	0	-	0	11		1	4	32	11	11	3s	6 827	177 593		0		0	0	0	0	*no check*		1/1	Y	-				-
Backend	0	0		0	11		1	4	20	11	11	3s	6 827	177 593	0	0		0	0	0	0	16m8s UP		1/1	1	0		0	0s	

stats

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	1	-	1	1	200	1			0	0	0	0	0					OPEN								
Backend	0	0		0	0		0	0	20	0	0	0s	0	0	0	0		0	0	0	0	16m8s UP		0/0	0	0		0		

Deployment – Docker Compose



compose-viz/compose-viz: A compose file visualization tool that follows compose-spec and allows you to gernerate graph in several formats.

Database

```ts
import { Client } from 'pg';
import dotenv from 'dotenv';

dotenv.config();

if(process.env.PG_URI === undefined){
    throw new Error( message: "DB URL is not defined via env variable")
}

const client = new Client({
    connectionString: process.env.PG_URI,
});

client.connect().then(r :void => console.log( message: `Connected to DB`)); Musa Rochi

export default client; Show usages Musa Rochi
```

```ts
interface TimeLog { Show usages Musa Rochi
    id: number;
    date: string;
    start: string;
    stop: string;
    break_duration: { hours: string; minutes: string };
    work_time: { hours: string; minutes: string };
}
```

```sql
CREATE TABLE IF NOT EXISTS logs (
    id SERIAL PRIMARY KEY,
    date DATE NOT NULL,
    start TIME NOT NULL,
    stop TIME NOT NULL,
    break_duration INTERVAL NOT NULL,
    work_time INTERVAL NOT NULL
);
```

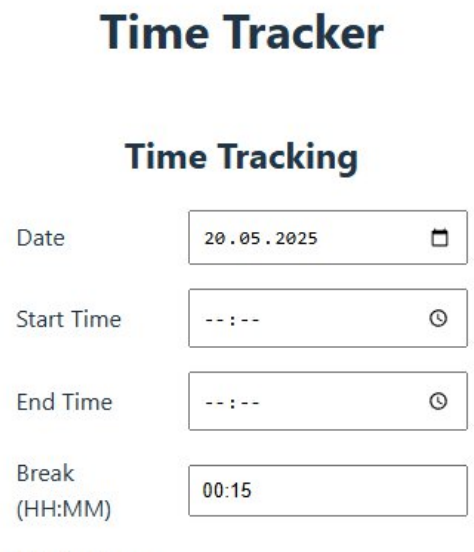
Backend

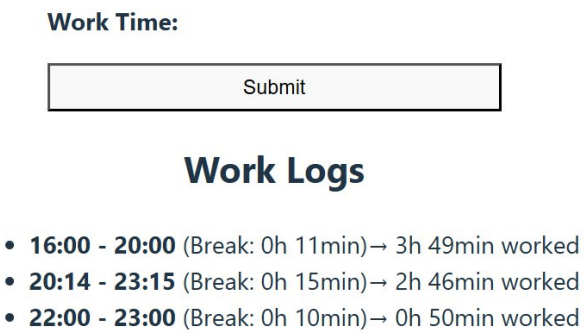
- NodeJS
- Express

```
router.post("/log", async (req: Request, res: Response): Promise<any> => {
    const {date, start, stop, break: breakDuration} = req.body;

    if (!date || !start || !stop || !breakDuration) {
        return res.status(400).json({error: "All fields are required!"});
    }

    const startDateTime = new Date(`${date}T${start}:00`);
    const endDateTime = new Date(`${date}T${stop}:00`);
    const [breakHours, breakMinutes] = breakDuration.split(":").map(Number);
    const breakMillis = (breakHours * 60 + breakMinutes) * 60 * 1000;

    const workTimeMillis = endDateTime.getTime() - startDateTime.getTime() - breakMi
    const workHours = Math.floor(workTimeMillis / (1000 * 60 * 60));
    const workMinutes = Math.floor((workTimeMillis % (1000 * 60 * 60)) / (1000 * 60)
    const workTime = `${workHours}:${workMinutes}`;

    const query = `INSERT INTO logs (date, start, stop, break_duration, work_time)
                   VALUES ($1, $2, $3, $4, $5) RETURNING *`;
    const values = [date, start, stop, breakDuration, workTime];

    try {
        const result = await client.query(query, values);
        const newLog = result.rows[0];

        eventClients.forEach(client => client(newLog));

        res.status(201).json({message: "Log saved successfully!", log: newLog});
    } catch (error) {
        console.error("Error saving log:", error);
        res.status(500).json({error: "Failed to save log."});
```

Frontend


React


Time Tracker
Time Tracking
Date
20.05.2025
Start Time
--:--
End Time
--:--
Break
(HH:MM)
00:15


Work Time:
Submit
Work Logs
16:00 - 20:00 (Break: 0h 11min)→ 3h 49min worked
20:14 - 23:15 (Break: 0h 15min)→ 2h 46min worked
22:00 - 23:00 (Break: 0h 10min)→ 0h 50min worked

Demo