INF142 Obligatorisk Oppgave 1

Nikolai Gangstø

Februar 2025

Innhold

Github

Svaret mitt finnes også på: `https://github.com/nikolaihg/INF142-Mandatory-1`.

a) Applikasjonsnivåprotokoll

1.1 Format

All kommunikasjon mellom klient og server skjer via tekststrenger som er formatert med semikolon (;) som skilletegn. Disse tekststrengene blir deretter prosessert av serveren og lagret i en database som et Python-dictionary.

Generelt format for klientmeldinger:

Klienten sender en streng som starter med et tall som representerer kommandoen, så etterfulgt av andre parameterer avhengig av hvilken kommando som blir gitt og ønsket respons. Er ikke noe grense på hvor mange alternativ du kan legge til, så lenge de kommer sist.

```
<kommando>;<problemID>;<tittel>;<alternativ1>;<alternativ2>;...
```

Eksempel for å opprette et problem:

```
1;100;Hva skal jeg ha til middag?;Pølse;Hamburger;Pizza
```

Python-databasen:

```
database = {
    100: {
        "tittel": "Hva skal jeg ha til middag?",
        "alternativ": {
            "Pølse": 0, "Hamburger": 0, "Pizza": 0
        }
    },
    (...)
}
```

1.2 Meldinger

1.2.1 Legg til problem (valg 1 i client.py)

Klienten sender en kommando for å legge til et nytt problem med tittel og alternativer. Serveren lagrer problemet i databasen og returnerer en bekreftelse.

- Klient → Server:

  ```
  "1;100;Hva skal jeg ha til middag?;Pølse;Hamburger;Pizza"
  ```

- Server → Klient:

  ```
  "Problem lagt til i databasen."
  ```

1.2.2 Vis alle problemer (valg 2 i client.py)

Klienten sender kommandoen "2" for å hente alle problemer fra databasen.

- Klient → Server:

```
"2"
```

- Server → Klient:

```
"{
    "100": {
        "tittel": "Hva skal jeg ha til middag?",
        "alternativ": {
            "Pølse": 0, "Hamburger": 0, "Pizza": 0
        }
    },
    "101": {
        "tittel": "Når skal jeg stå opp?",
        "alternativ": {
            "13:00": 0, "07:00": 0, "09:00": 0
        }
    }
}"
```

1.2.3 Vis et spesifikt problem (valg 3 i client.py)

Klienten sender kommandoen "3" sammen med problemID for å hente informasjon om et spesifikt problem.

- Klient → Server:

```
"3;100"
```

- Server → Klient (Hvis funnet):

```
"{
    "tittel": "Hva skal jeg ha til middag?",
    "alternativ": {
        "Pølse": 0, "Hamburger": 0, "Pizza": 0
    }
}"
```

- Server → Klient (Hvis ikke funnet):

```
"Feil: Problem med ID 100 finnes ikke."
```

1.2.4 Vis alternativer for et problem (valg 4 i client.py)

Klienten sender kommandoen "4" sammen med problemID for å hente alternativene for et spesifikt problem.

- Klient → Server:

```
"4;100"
```

- Server → Klient:

```
"{
    "Pølse": 0, "Hamburger": 0, "Pizza": 0
}"
```

1.2.5 Stem på et alternativ (valg 5 i client.py)

Klienten sender kommandoen "5" sammen med problemID og stemmen for å registrere en stemme.

- Klient → Server:

```
" 5;100;Pølse"
```

- Server → Klient:

```
"Stemme registrert: 'Pølse' for problem 100."
```

- Klient → Server:

1.2.6 Vis stemmer for et problem (valg 6 i client.py)

Klienten sender kommandoen "6" sammen med problemID for å hente stemmene for et spesifikt problem.

- Klient → Server:

```
6;100
```

- Server → Klient:

```
"{
    "Pølse": 1, "Hamburger": 0, "Pizza": 0
}"
```

1.3 Avslutning (valg 7. i client.py

Klienten kan avslutte forbindelsen ved å sende kommandoen "exit".

- Klient → Server:

```
"exit"
```

- Server → Klient:

```
(ingen respons, forbindelsen avsluttes)
```

b) Tilstandsløs vs tilstandsfull og TCP vs UDP

2.1 Tilstandsløs vs tilstandsfull

Applikasjonsprotokollen er tilstandsløs, siden hver melding inneholder all nødvendig informasjon.

2.2 TCP vs UDP

TCP er best egnet da vi trenger pålitelig overføring av data for å sikre at stemmer registreres riktig.

c) Implementering av klient-server-applikasjonen i Python

3.1 Se `server.py` og `client.py`.

NB: Jeg har ikke klart å implementere `post_problem`.