Projektna dokumentacija

1.1 Opis problema

Opredelitev problema

Problem pokrivanja košev (bin covering) je temeljni optimizacijski izziv, pri katerem je treba predmete različnih velikosti učinkovito razporediti v koše s fiksno kapaciteto. V nasprotju s tradicionalnim problemom pakiranja košev (bin packing), ki si prizadeva zmanjšati število uporabljenih košev, se pokrivanje posod osredotoča na maksimiziranje izkoriščenosti razpoložljivega prostora v posodah. Zaradi pomanjkanja raziskovalnih orodij za ta specifičen problem sem ustvaril interaktivno izobraževalno orodje, ki uporabnikom omogoča vizualizacijo, implementacijo in primerjavo različnih strategij pokrivanja posod v realnem času.

Cilji

Glavni cilj je ustvariti spletno interaktivno okolje, ki služi več namenom:

- **Hitro prototipiranje**: omogoča hitro implementacijo in testiranje algoritmov po meri.
- **Generiranje podatkov**: nudi orodja za ustvarjanje testnih naborov podatkov z nastavljivimi parametri.
- **Izobraževalno orodje**: ustvari dostopno platformo za študente in raziskovalce za učenje konceptov pokrivanja posod.
- **Primerjava strategij**: omogoča neposredno primerjavo različnih algoritemskih pristopov.

Ciljni uporabniki:

- **Študenti**: učenje algoritemskih konceptov in optimizacijskih tehnik.
- **Raziskovalci**: prototipiranje novih algoritmov za pokrivanje posod in primerjava njihove uspešnosti.
- **Izobraževalci**: poučevanje optimizacije in načrtovanja algoritmov.

1.2 Analiza in načrtovanje sistema

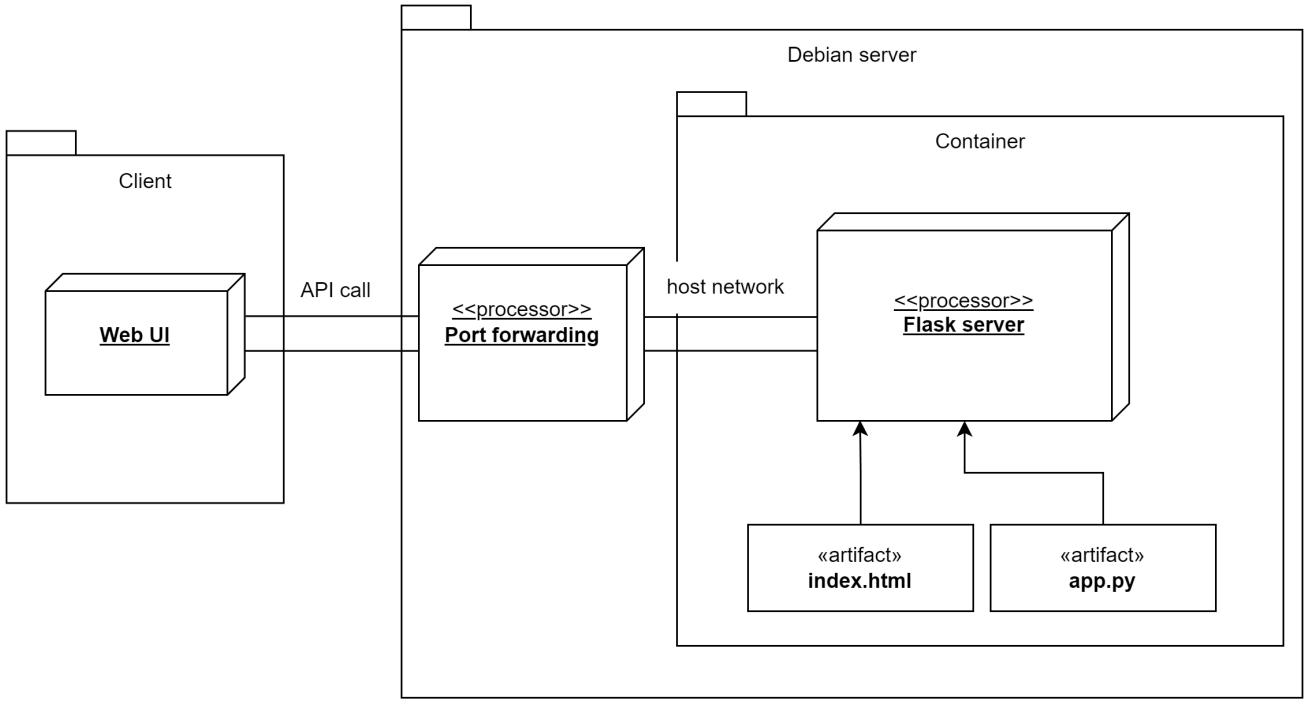
Funkcionalna specifikacije

- Omogočiti uporabniku pisanje Python kode v urejevalniku:
  - shranjevanje in nalaganje skript
  - poudarjanje skladnje in/ali napak.

- Omogočiti uporabniku organizacijo kode:
  - kategorizirano shranjevanje za dnevnike, algoritme, skripte
  - enostaven vmesnik z abstraktnimi razredi, ki omogočajo kakovostno in standardizirano kodo.
- Omogočiti uporabniku izvajanje in testiranje algoritmov/generatorjev:
  - ukazi, izvedljivi znotraj spletnega terminala
  - podpora za standardni vhod/izhod programa
  - prikazovanje napak, ki uporabniku pomagajo pri odpravljanju le-teh.
- Omogočiti uporabniku vizualizacijo kode:
  - vizualna predstavitev uvozov, razredov, metod in spremenljivk.

Arhitektura sistema

Sistem sledi arhitekturi odjemalec-strežnik (angl. client - server) z jasno ločitvijo nalog. UML diagram namestitve (angl. deployment diagram) pomaga pri razumevanju arhitekture projekta:



Objektno usmerjeni načrtovalski vzorci

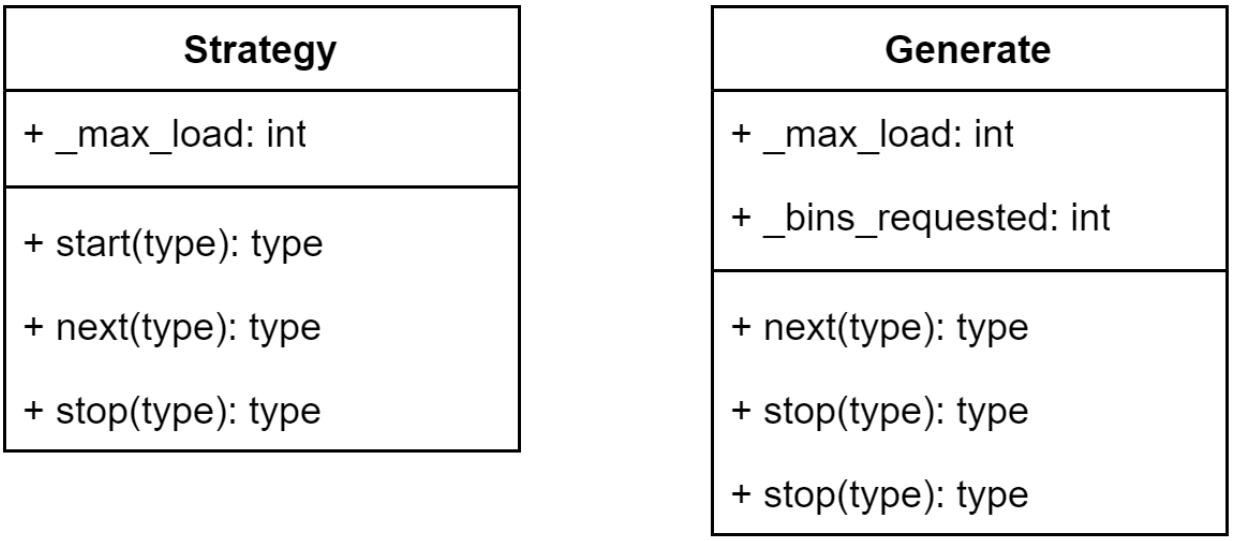
Implementacija načrtovalskega vzorca Strategija (Strategy pattern): Sistem implementira vzorec strategije preko abstraktnih osnovnih razredov:

- `Strategy` : Abstraktni osnovni razred, ki definira vmesnik za algoritme pokrivanja posod.
- `Generate` : Abstraktni osnovni razred za algoritme generiranja predmetov.

Načrtovalski vzorec Predloga (Template method pattern): Oba osnovna razreda definirata predlogo za izvajanje algoritma:

1. **Inicializacija**: Faza nastavitve s konfiguracijskimi parametri.

2. **Začetek**: Začetek izvajanja algoritma z začetnim stanjem.
3. **Naslednji**: Iterativna obdelava posameznih predmetov.
4. **Konec**: Zaključevanje in poročanje o rezultatih.
   Naslednja slika predstavlja UML diagram razredov za naše predloge:



Modularna arhitektura

Ločevanje komponent:

- **app.py**: Flask strežnik in definicije API končnih točk.
- **main.js**: Logika uporabniškega vmesnika (frontend) in upravljanje interakcij z uporabnikom.
- **script-visualizer.js**: Pogon za vizualizacijo kode.
- **style.css**: Oblikovanje uporabniškega vmesnika in definicije teme.

Organizacija direktorijev in datotek:

```
web_interface/
├── src/
│   ├── generate_bins/      # Algoritmi za generiranje predmetov
│   ├── strategies/         # Strategije pokrivanja posod
│   ├── custom_code/        # Uporabniško definirane skripte
│   └── data/               # Generirani nabori podatkov in dnevniki
├── static files (HTML, CSS, JS)
└── base classes (strategy.py, generate.py)
```

1.3 Izvedba

Izbrana orodja in tehnologije

Tehnologije uporabniškega vmesnika (Frontend)

HTML5/CSS3 - spletni standard.
Javascript - za upravljanje logike na uporabniškem vmesniku in klice API-jev.
Urejevalnik Ace - urejevalnik kode, ki ponuja:

- Poudarjanje skladnje za Python z različnimi prilagoditvami.
  XTerm.js - terminalski emulator, ki deluje kot dodatek za `html` in `js` aplikacije.

Ogrodje zalednega sistema (Backend)

Flask - lahko spletno ogrodje za Python, izbrano zaradi:

- Hitrega razvoja in enostavne namestitve.
- Zmožnosti razvoja RESTful API-jev.
- Vgrajenega razvojnega strežnika.

Modul `subprocess` - za varno izvajanje skript:

- Izolirano okolje za izvajanje.
- Nadzor nad časovnimi omejitvami za večjo varnost.
- Zajem izhodnih podatkov in tokov napak.

Razvoj projekta

Upravljanje različic z Gitom:

- Projekt gostuje na `https://github.com/jakobb01/BinCovering`.
- Sledenje spremembam (commitom) in zgodovina različic.
- Podpora za sodelovalni razvoj.
- Sledenje težavam (issues) in dokumentacija.

Kontejnerizacija z Dockerjem za:

- Konsistentno okolje za namestitev.
- Osnovna slika (image) Python 3.11 slim.
- Upravljanje odvisnosti za Flask.
- Izpostavitev vrat 5000 za spletni dostop.