Izvještaj o aplikaciji SmartHouse

Skyline talent program

Ime i prezime: Amer Mujalo

Broj indexa: 19101

Datum: 09.02.2025

Projektni Izvještaj

Pregled promjena i unapređenja

Ovaj dokument sadrži detaljan pregled svih commit-a od 2. do 9. februara 2025. godine, uključujući implementacije novih funkcionalnosti, refaktorisanje koda i optimizacije sistema.

---

2. februar 2025.

feat: [Backend] Implementacija klase SmartHouse i dodavanje naslijeđenih klasa Device i Room

- Dodana klasa **SmartHouse** koja upravlja uređajima i sobama.
- Naslijeđene klase **Device** i **Room** omogućavaju bolju organizaciju objekata.
- Prerađen **Program.cs** kako bi bolje simulirao rad aplikacije.
- Razmatra se mogućnost uklanjanja naslijeđenih klasa iz **Room**, jer njihova dodatna apstrakcija nije neophodna.

---

3. februar 2025.

feat: [Backend] Implementacija UI konzolne aplikacije i dorade SmartHouse klase

- Dodana interaktivna konzolna aplikacija za upravljanje sobama i uređajima.
- Uvedene osnovne funkcionalnosti za dodavanje i upravljanje komponentama.
- Planirano restrukturiranje hijerarhije klasa **Room** i **SmartHouse** radi optimizacije.

refactor: [Backend] Restrukturiranje hijerarhije klasa i uklanjanje redundantnih klasa

- **SmartHouse** klasa uklonjena zbog suvišnosti.
- **Room** i **SmartHouse** spojeni u jednu univerzalnu klasu **Objekat**.
- **Objekat** sada može predstavljati i kuću i sobu, čime se smanjuje kompleksnost koda.

refactor: [Objekat.cs] Optimizacija metode pretrage komponenti

- Spajanje metoda **NadjiSobu** i **NadjiUredjaj** u jedinstvenu **NadjiKomponentu**.
- Omogućeno generičko pretraživanje komponenti bilo kojeg tipa unutar objekta.

---

4. februar 2025.

feat: [Objekat.cs] Dodane metode MoveTo i NadjiParentUredjaj

- **MoveTo** omogućava premještanje uređaja između soba.
- **NadjiParentUredjaj** olakšava pretragu nadređenog uređaja unutar hijerarhije objekata.

fix: [Objekat.cs, ASmartComponent.cs] Sprečavanje kreiranja beskonačnih petlji

- Implementirana detekcija ciklusa pri dodavanju novih komponenti.
- Blokirano dodavanje čvorova koji su već povezani u hijerarhiji kako bi se spriječila beskonačna rekurzija.

---

5. februar 2025.

refactor: [Objekat.cs, ASmartComponent.cs] Završena refaktorizacija, spremno za merge u develop

- Optimizacija hijerarhijske strukture kodova.
- Premještanje dijela funkcionalnosti iz **Objekat.cs** u **ASmartComponent.cs** radi bolje modularnosti.
- Potpuno riješen problem hijerarhijske cikličnosti.

refactor: [composite/] Čišćenje i optimizacija za merge u main

- Uklonjene zastarjele funkcionalnosti.
- Poboljšana čitljivost i održivost koda.
- Projekt je spreman za spajanje u **main** granu.

---

7. februar 2025.

refactor: [composite/] Dodavanje podklasa specifičnih za sobe i unapređenje hijerarhijskih pravila

- Dodane specifične podklase soba radi bolje organizacije.
- Poboljšana pravila za upravljanje hijerarhijom objekata.

---

8. februar 2025.

test: [SmartHouseTests] Dodan testni projekat sa 40 unit testova

- Implementiran **SmartHouseTests** projekat.
- Pokriveno 40 unit testova za verifikaciju ispravnosti sistema.
- Poboljšana testna infrastruktura.

---

9. februar 2025.

refactor: [Program.cs, Objekat.cs] Poboljšano UI upravljanje i unaprijeđeno listanje komponenti, spremno za merge u Main

- Optimiziran interfejs za interaktivno upravljanje objektima.
- Poboljšana prezentacija podataka u konzoli.
- Pripremljeno za spajanje u **Main** granu.

feat: [Objekat.cs] Added power consumption tracking and implemented DeepRemove method for removing components from the top of the tree

- Dodano praćenje potrošnje energije (`powerConsumption`).
- Implementirana metoda `DeepRemove & DeepRemoveID` za uklanjanje komponenti sa vrha stabla.
- Osigurano pravilno ažuriranje potrošnje energije prilikom dodavanja i uklanjanja komponenti.

Dodatni Issue-i

Issue #1: Da li su posebne klase za sobe (SpavaćaSoba, DnevnaSoba...) potrebne?

Opis:
Trenutno razmatram da li su mi posebne klase za sobe, poput **SpavaćaSoba** i **DnevnaSoba**, zaista potrebne. Ove klase nasljeđuju **Objekat** (ranije **Soba**), ali njihova svrha je upitna jer se funkcionalnosti kuće ostvaruju dodavanjem uređaja na instancu **Objekat**.
Klasa **Objekat** već sadrži listu komponenti koje mogu biti uređaji (**Device** tipovi) ili drugi objekti (**Objekat** tipovi), što znači da se hijerarhija prostora već može formirati bez posebnih klasa za različite tipove soba.
Pitanja za razmatranje:

1. Postoje li funkcionalnosti koje bi opravdale postojanje posebnih klasa za sobe?
2. Da li se razlikovanje soba može riješiti putem atributa umjesto nasljeđivanja?
3. Postoje li scenariji gdje bi nam specifične sobe donijele korist u strukturi koda?

Rješenje:
Dodali smo dodatne klase koje nasljeđuju **Objekat**, a to su **Kuca**, **Sprat** i **Soba**. Njihova glavna funkcija je da kontrolišu strukturu objekata u sistemu, odnosno da zabrane korisniku dodavanje nepravilnih komponenti.

- **Kuca** može sadržavati samo **Sprat** ili **Device**.
- **Sprat** može sadržavati **Sobu** ili **Device**.
- **Soba** može sadržavati isključivo **Device**.

Na ovaj način osigurali smo logičnu hijerarhiju elemenata unutar sistema.
Što se tiče specifičnih klasa za sobe (**SpavaćaSoba**, **DnevnaSoba**...), ostavili smo ih u projektu, iako trenutno nemaju neku posebnu funkcionalnost. Glavna klasa koja upravlja ovom strukturom ostaje **Objekat.cs**.
Ove promjene smo implementirali tek nakon commita:
📌 "Added SmartHouseTests project with 40 tests and improved testing functionality".

Issue #2: Provjera poboljšanja koda nakon spajanja klasa SmartHouse i Soba u klasu Objekat

Opis:
Nedavno sam izvršio refaktorisanje koda spajanjem klasa **SmartHouse** i **Soba** u novu klasu **Objekat**, koja sada predstavlja čvor u stablu. Ovim smo eliminisali duplirani kod i smanjili redundantnost, ali smo izgubili mogućnost da imamo posebnu klasu za kuću.

Potencijalni problem koji se sada javlja je to što više nemamo mehanizam da spriječimo dodavanje kuće u sobu, tj. kod poput `soba.Add(kuca)` sada nije eksplicitno zabranjen.

Pitanja za razmatranje:

1. Da li je kod sada zaista čistiji i lakši za održavanje?
2. Da li je potrebno uvesti dodatne mehanizme za validaciju strukture stabla, kako bismo spriječili nepravilne odnose između objekata?
3. Postoji li bolji način da riješimo ovaj problem bez vraćanja na prethodnu strukturu?

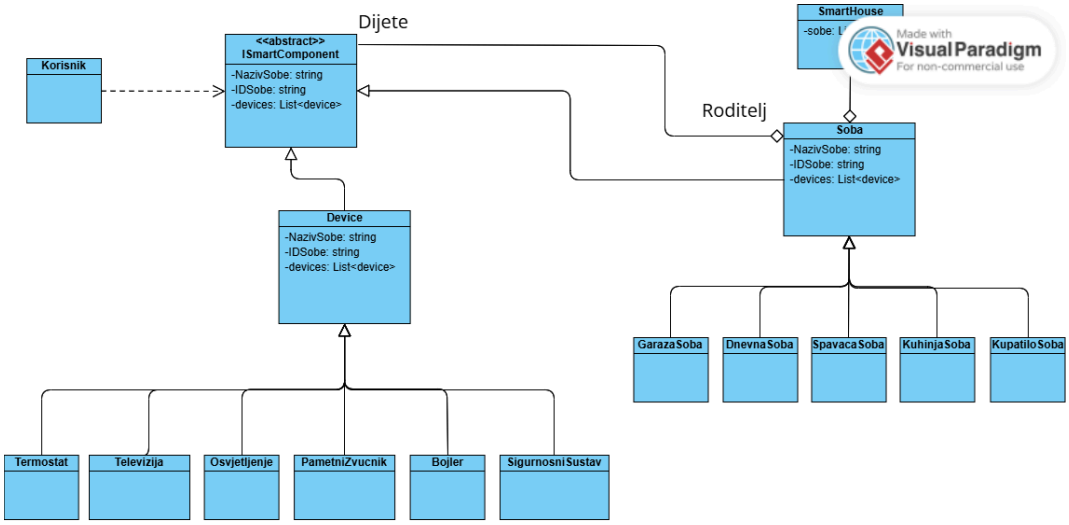
Rješenje:

1. **Da li je kod sada zaista čistiji i lakši za održavanje?**
    - Da, refaktorisanje je donijelo značajnu fleksibilnost i skalabilnost. Sada možemo modelirati različite hijerarhijske strukture, ne samo kuće sa sobama, već i kompleksnije objekte poput zgrada sa spratovima i stanovima, a da pri tome ne dupliramo kod. Ovaj pristup omogućava bolje iskorištavanje zajedničkih funkcionalnosti i olakšava buduća proširenja sistema.
2. **Da li je potrebno uvesti dodatne mehanizme za validaciju strukture stabla?**
    - Da, i to je već riješeno u grani **bugfix/hierarchical-cycle-prevention**. Implementirali smo provjeru koja osigurava da se objekat ne može dodati unutar samog sebe niti unutar svog potomstva, čime smo spriječili nepravilne odnose između čvorova u stablu. Ovim mehanizmom garantujemo validnost hijerarhije objekata.
3. **Postoji li bolji način da riješimo ovaj problem bez vraćanja na prethodnu strukturu?**
    - Jedan od mogućih pristupa mogao bi biti uvođenje različitih tipova objekata putem polimorfizma ili označenih tipova (tagged types). Na primjer, mogli bismo imati enumeraciju ili polje tipa **ObjectType** unutar klase **Objekat** koje jasno označava da li je neki objekat kuća, soba, sprat, stan itd. Prilikom dodavanja novih čvorova mogli bismo provjeravati da li je određena kombinacija dozvoljena (npr. sprat može sadržavati sobe, ali soba ne može sadržavati spratove).

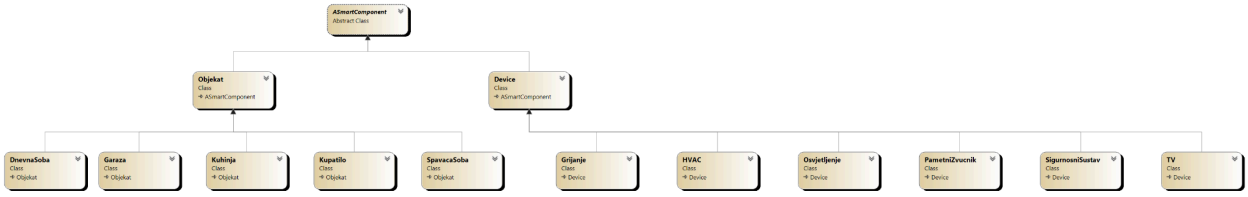
Tako bismo dobili dodatni sloj validacije bez vraćanja na prethodnu strukturu, a istovremeno bismo zadržali fleksibilnost trenutnog rješenja.

Class Diagrami

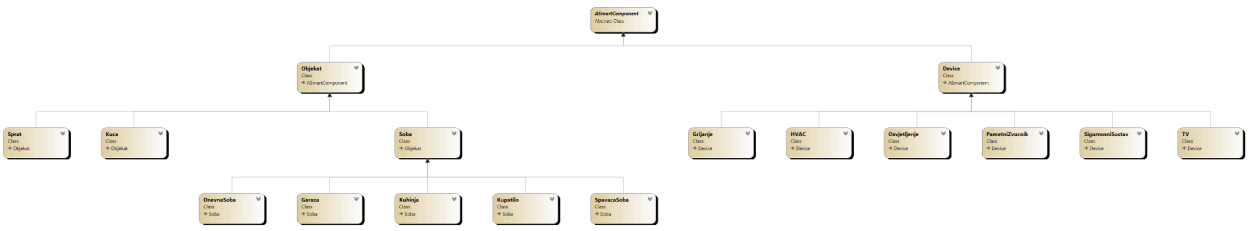
1. **Demo Class Diagram**
   Ovo je inicijalni class diagram koji sam samostalno nacrtao kako bih prikazao osnovnu strukturu projekta prije refaktorisanja.



2. **SmartHouseV2 Class Diagram**
   Automatski generisan class diagram za verziju **SmartHouseV2** projekta. Ova verzija uključuje klase **SmartHouse**, **Room**, i **Device** sa osnovnim hijerarhijama.



3. **SmartHouseV3 Class Diagram**
   Automatski generisan class diagram za verziju **SmartHouseV3** projekta. Ova verzija uključuje klasu **Objekat** koja je rezultat refaktorisanja i spajanja klasa **SmartHouse** i **Room**. Takođe, uključuje i nove klase **Kuca**, **Sprat**, i **Soba** koje su dodane radi bolje kontrole hijerarhije. **Ovo sve isto vazi i za V4**.



Zaključak

Tokom prethodnih sedmica implementirane su ključne funkcionalnosti, izvršena su značajna poboljšanja strukture koda i dodani su neophodni testovi. Projekat sada ima stabilniju arhitekturu, poboljšane funkcionalnosti i bolju modularnost. Sve izmjene su spremne za završni merge u **Main** granu.