Grupa 1

LightsOn
Smjernice za dizajn

Verzija 1.0

Istorija revizija

Datum	Verzija	Opis	Autor
11.11.2024.	1.0	Inicijalna verzija dokumenta	Aleksej Mutić

Sadržaj

Smjernice za dizajn

1. Uvod

Dokument smjernice za dizajn daje detaljan opis standarda arhitektonskog dizajna, kao i konvencije dizajna, kojim se obezbjeđuje efikasna implementacija sistema.

1.1 Svrha

Svrha dokumenta je opis standarda za dizajn, kao i konvencija i idioma koji se koriste u procesu dizajna sistema.

1.2 Područje primjene

Dokument predstavlja sponu za razmjenu informacija između različitih dijelova razvojnog tima, koji su ključni sudionici u procesima planiranja arhitekture i implementacije sistema. Samim tim olakšava usklađivanje rada i sprječava probleme nastale u toku realizacije, uzrokovane različitim poimanjem sistema. Novim članovima tima omogućava razumijevanje o tome šta se proizvodi.

1.3 Definicije, akronimi i skraćenice

U sljedećoj tabeli navedene su definicije pojmova korišćenih u ovom dokumentu.

JavaDoc	Proširivi sistem za generisanje dokumentacije koja čita posebno formatirane komentare u Java izvornom kodu i generiše kompajliranu dokumentaciju. Obično se koristi za izradu API dokumentacije u obliku HTML web stranica
Garbage collector	Sistem automatizovanog upravljanja memorijom, tj. proces kojim Java briše nekorišćene objekte iz memorije
JFC(Java Foundation Classes)	Skup biblioteka u Javi koje omogućavaju razvoj grafičkih korisničkih interfejsa (GUI) za desktop aplikacije

Sve potrebne definicije, akronimi i skraćenice sadržane su u dokumentu *Rječnik*, koji je dio projektne dokumentacije.

1.4 Reference

[1] *Projektovanje i dizajn softvera: koncepti, principi i projektni obrasci*, prof. dr Vladimir Vujović
[2] *Software Design Principles and Guidlines*, Douglas C. Schmidt

1.5 Pregled

Dokument Smjernice za dizajn sačinjen je od sljedećih dijelova:

- Uvod, koji daje informacije o svrsi dokumenta i području primjene
- Opšte smjernice za dizajn i implementaciju, gdje su opisani principi i strategije korišćene u procesima dizajna i implementacije sistema
- Smjernice za dizajn baze podataka, gdje su data pravila i preporuke dizajna baze podataka
- Smjernice za dizajn arhitekture, gdje su data pravila i preporuke dizajna arhitekture softvera
- Smjernice za mehanizme, gdje su opisani mehanizmi nižih slojeva sistema

2. Opšte smjernice za dizajn i implementaciju

2.1 Mapiranje dizajna na implementaciju

Dizajn sistema nastaje postupno. Na osnovu analize zahtjeva zainteresovanih strana, kreira se model zahtjeva čiji su funkcionalni zahtjevi direktno predstavljeni dijagramima slučajeva korišćenja. Domenski i klasni dijagram nastaju oslanjajući se na dijagram slučajeva korišćenja. Model podataka, pogled i kontroler su usklađeni implementacijom Komanda projektnog obrasca (Command). Samim tim kontroler protumačene akcije korisnika delegira kao komande modelu. Na osnovu prethodnog dobija se bolje struktuiran MVC model, koji pruža laku proširivost i labavu spregu, kao i bolju razumljivost arhitekture. Modelovanje se vrši u softverskom alatu SAP

PowerDesigner, koji omogućava kreiranje osnovnih paketa MVC modela (model, view, controller), kao i detaljno specificiranje potrebnih klasa, interfejsa, modela, i veza između njih. U procesu implementacije navedeni elementi se prevode u istoimene pakete sa pripadajućim klasama u Eclipse razvojnom okruženju. Pored uvezenih paketa, radi bolje strukturiranog koda kreiraju se dodatni podpaketi. Paket observer izdvaja i pojašnjava primjenu istoimenog projektnog obrasca.

2.2 Specifikacija interfejsa za podsisteme

Paketi, odnosno klase koje paketi posjeduju se projektuju sa ciljem da budu autonomni, što omogućava modularnost i iskoristivost, kojom se postiže labava sprega između klasa. Akcent je stavljen na interfejse kao tačke komunikacije između klasa, pri čemu je očuvana njihova saradnja nezavisno od implementacije. Samim tim promjene na postojećim klasama neće nametati izmjene na ostatak sistema. Tako u paketu view klase/interfejsi kreiraju izgled softverskog alata, koji je nezavisan od potencijalnih nadogradnji modela.

2.3 Dokumentovanje koda

Adekvatna dokumentacija svih ključnih elemenata sistema osigurava laku održivost sistema. Zbog toga je JavaDoc dokumentacija koda obavezna, odnosno dokumentovanje svih klasa i metoda, što uključuje opis klase/metode, navođenje povratnih vrijednosti (ukoliko postoje), parametara metoda kao i naziv autora metode/klase. U toku izrade softvera, moguće je koristiti inline komentare, kako bi se programeru olakšala implementacija. Pravila imenovanja elemenata u kodu su obrađena u dokumentu *Smjernice za programiranje*.

2.4 Otkrivanje, rukovanje i izvještavanje o greškama i izuzecima

U toku projektovanja softvera neophodno je obezbjediti mehanizme za otkrivanje i rukovanje greškama, kao i oporavak sistema, bez prekida u njegovom radu. Najjednostavnije rješenje za rukovanje izuzecima predstavlja try-catch blok, koji se realizuje obmotavanjem dijela koda na kojem se očekuje izuzetak, uz adekvatne rukovaoce izuzecima unutar catch bloka.

2.5 Upravljanje memorijom

Za upravljanje memorijom zadužen je Garbage collector programskog jezika Java. Bitna stavka za ispravno upravljanje memorijskih resursa, kao i realizovanje što boljih performansi sistema jeste odabir adekvatne arhitekture. MVC arhitektura pri modelovanju obezbjeđuje smanjenje redudantnosti podataka i efikasno korišćenje resursa.

2.6 Distribucija softvera

Softverski alat LightsOn je moguće preuzeti sa zvanične stranice na internetu.

2.7 Ponovna iskoristivost komponenti

Uzevši u obzir korišćenu MVC arhitekturu, sačinjenu od tri različite nezavisne komponente, omogućeno je korišćenje podataka modela za formiranje različitih prikaza. Samim tim je i omogućeno projektovanje sistema prilagođeno različitim okruženjima.

2.8 Posebna upotreba jezičkih karakteristika

Tokom dizajna softvera ne navode se jezička ograničenja i restrikcije, u cilju projektovanja sistema koji je nezavisan od programskog jezika koji će se koristiti za implementaciju.

2.9 Struktura projekta

Projekat je podijeljen u logičke cjeline predstavljene paketima. Paketi će se nalaziti u roditeljskom paketu *source*, odnosno predstavljaće njegove podpakete. Pored toga potrebno je omogućiti skladištenje korišćenih resursa, kao što su slike, ikonice i sl. u posebne direktorijume.

2.10 Smjernice za razvoj i izmjene sistema

Praćenje preporuka u datom dokumentu, projektnom timu je omogućena izrada softverskog alata koji je lako proširiv, pogodan za izmjene uz minimalan napor, a pored toga modelovan da podrži

implementaciju i u drugim programskim jezicima i tehnologijama.

3. Smjernice za dizajn baze podataka

Softverski alat LightsOn ne koristi sopstvenu bazu podataka. Kreirani projekti i modeli skladišteni su na sekundarnoj memoriji računara.

4. Smjernice za dizajn arhitekture

U skladu sa karakteristikama problema koji se rješava, za projektovani sistem izabrana je MVC (Model – View - Controller) arhitektura, kojom je omogućena održivost i modularnost sistema i obezbijeđeni različiti načini interakcije i prikaza podataka u sistemu, koji su nezavisni od promjena u toku životnog ciklusa. Odabirom MVC arhitekture sistem je struktuiran u tri logičke cjeline, čime je osigurana nezavisnost i razdvajanje podataka i njihovog prikaza. Model sadrži poslovnu logiku i upravlja podacima. Implementiran je kao aktivni model, kroz mehanizam ažuriranja i obavještavanja pogleda i kontrolera o izmjenama podataka. Navedeni mehanizam je implementiran korišćenjem Posmatrač projektnog obrasca (Observer).

5. Smjernice za mehanizme

Pri izradi softverskog alata razvojni tim može koristiti standardne metode i mehanizme implementirane kroz Java programski jezik, preciznije JFC biblioteku Swing. Upotreba alata neće biti problem za korisnike sa osnovnim poznavanjem rada na računaru. U slučaju poteškoća pri korišćenju dostupno je korisničko uputstvo i online pomoć, gdje su ponuđena detaljna objašnjenja o odgovarajućim segmentima aplikacije, kao i često postavljena pitanja.