TECHNIESE ONTWERP

Planning Generator

Abdi Ali

Inhoud

Versie beheer

Versie:	Datum:	Door wie:	Waar:
1.00.0	02-10-2022	Abdi Ali	Hele document
1.00.2	17-10-2022	Abdi Ali	Hele document
1.00.3	20-10-2022	Abdi Ali	activity
1.00.4	4-12-2022	Abdi ali	Klassen diagram

Inleiding

In dit technische ontwerp zullen we ons richten op de ontwikkeling van een systeem voor Randstad, een groeiend uitzendbureau dat veel nieuwe werknemers inhuurt. Dit systeem zal gericht zijn op het automatiseren van het proces van planning en roostering van personeel voor productiebedrijven. We zullen ons concentreren op hoe het systeem de wensen van zowel productiebedrijven als personeelsleden kan aanpassen en hoe het problemen die ontstaan bij de handmatige planning kan oplossen. Hierdoor zal het systeem zorgen voor een efficiëntere en effectievere planning en roostering van personeel.

Samenvatting

Dit document beschrijft de technische aspecten van het systeem planning generator. Hierin worden de technische specificaties uit het functioneel ontwerp geanalyseerd, waaronder de systeemarchitectuur, gebruikte databases, algoritmische structuur, class diagram en overwegingen. Hierdoor wordt een duidelijk beeld geschetst van hoe het systeem eruit zal zien en functioneren, inclusief de relaties tussen de verschillende onderdelen en componenten van het systeem en de overwegingen die genomen zijn bij de implementatie van de algoritmische structuur en class diagram.

Probleemstelling

Door de handmatige aanpak die Randstad gebruikt bij het creëren van planningen, komen er vaak fouten voor. Op basis hiervan is besloten om een systeem te ontwikkelen dat deze processen automatiseert en optimaliseert.

Wat wordt er gerealiseerd?

Er wordt een planning generator ontwikkeld om de taak van het plannen van werkzaamheden voor fysieke planners te automatiseren. Hiervoor wordt een algoritme geïmplementeerd dat in staat is om de wensen van de betrokken partijen te analyseren en een passende planning te genereren.

Activity diagram

Omdat het systeem een hoge algoritmische complexiteit heeft, wordt er in dit document een algoritmisch schematisch ontwerp weergegeven in de vorm van een activity diagram. Dit diagram legt de nadruk op de voorwaarden, beperkingen en de volgorde van stappen die zijn opgenomen in het algoritme (zie Bijlage Activity diagram.pdf). Dit geeft een duidelijke visuele weergave van de logica en de flow van het algoritme, en helpt bij het begrijpen van de technische details van het systeem.

Klassen Diagram

In dit deel van het document wordt de logische structuur van het systeem (back-end) weergegeven in een klassendiagram (zie bijlage "klassendiagram.pdf").

Data opslag

Aangezien de focus van de applicatie ligt op de functionaliteiten, is de dataopslag eenvoudig opgezet. De data wordt opgeslagen in een lokale binary file waarin geschreven en gelezen kan worden. Omdat de applicatie in een N-tier architectuur is opgezet, zullen latere wijzigingen in de dataopslag gemakkelijk kunnen worden aangebracht zonder dat dit invloed heeft op de rest van de architectuur. (zie Bijlage voor informatie over de architectuur)

Systeem beheer

In dit gedeelte zal ik beschrijven welke software en hardware er worden gebruikt voor het systeem, inclusief de specificaties van deze componenten.

Software

Software	
Programmeer talen	C# .net 6, html5 bootstrap5
database	mysql Verie: 5.2.0
Testomgeving/Containers	Docker 4.7 0
Ide	Visual studio 2022
repository	Github 3.6.2

Architectuur

Inleiding

De applicatie is opgebouwd in een N-tier architectuur, een structuur waarbij de applicatie bestaat uit meerdere lagen. In dit geval zal de applicatie drie lagen bevatten: de prestatie, domein en data-accesslaag. Hieronder zal ik voor elke laag een korte beschrijving geven van het doel ervan, evenals een voorbeeld van de flow van het object door de lagen heen en de redenen waarom ik voor deze architectuur heb gekozen.

Prestatie

het bovenste niveau van de applicatie is de gebruikersinterface. De hoofdfunctie van deze laag is om taken te vertalen naar een vorm die de gebruiker begrijpt en om de resultaten van die taken aan de gebruiker weer te geven.

Domein

Deze laag coördineert de toepassingen, verwerkt opdrachten, maakt logische beslissingen, evalueert en uitvoert berekeningen. Het communiceert ook met de omringende lagen, verwerkt gegevens en verzendt deze tussen de lagen.

DAL

In deze laag wordt informatie opgeslagen en opgehaald vanuit een database of bestandssysteem. De informatie wordt vervolgens doorgestuurd naar de logische laag via de prestatie laag voor verwerking en dan uiteindelijk terug naar de presentatie laag voor weergave aan de gebruiker.

Waarom Deze Architectuur?

Ik heb gekozen voor een n-tier architectuur omdat het de onder houdbaarheid van de applicatie verbetert. Door deze architectuur kan een van de lagen eenvoudig worden vervangen door een nieuwe laag, bijvoorbeeld als de opdrachtgever wil overschakelen van een webapp naar een Android-app. In dit geval kan een nieuwe presentatie laag worden gemaakt die de Android-app UI heeft en worden samengevoegd met de andere lagen, zonder dat de logische of data-accesslaag aangepast hoeft te worden. Dit principe kan ook worden toegepast op de data-accesslaag, bijvoorbeeld wanneer de gebruiker wil veranderen van database.

Dit is de reden waarom ik heb gekozen voor een n-tier architectuur.

Voorbeeld

Zoals te zien is in afbeelding Architectuur.pdf , heeft de domeinlaag geen directe verbinding met de andere lagen, maar weten de andere lagen wel dat de domeinlaag bestaat. Dit is vooral handig voor de data-accesslaag, waardoor er geen DTO-classes hoeven te worden gemaakt voor de classes die opgehaald moeten worden. Echter ontstaat er wel een probleem doordat de domeinlaag nu niet weet wat de data-accesslaag heeft. Dit probleem is opgelost door een interface te maken die gevuld

wordt in de prestatie-laag en wordt doorgegeven aan de domeinlaag.

Overwegingen

In deze alinea zullen we de overwegingen bespreken die zijn genomen bij het ontwerpen en bouwen van de planning generator. Hierbij zullen we specifiek ingaan op de uitdagingen die zijn ontstaan bij het automatiseren van het planningsproces, de keuzes die zijn gemaakt voor de technologieën en architectuur die zijn gebruikt, en de afwegingen die zijn gemaakt tussen efficiëntie en gebruiksvriendelijkheid. Door deze overwegingen te bespreken, zullen we een beter begrip krijgen van hoe het systeem is opgebouwd en hoe het zal functioneren in de praktijk.

Architectuur

Architectuur: Zoals beschreven in het vorige hoofdstuk, heb ik een voorbeeld gegeven van hoe ik de architectuur heb opgebouwd. De methode die ik heb gebruikt heeft zijn voordelen, maar een nadeel is dat de Data-accesslaag weliswaar via de prestatie-laag kan worden benaderd. Een overweging hierbij is dat een lijn wordt getrokken tussen de lagen en er gebruik wordt gemaakt van DTO's (Data Transfer Objects) Hierbij wordt de benodigde data van de Data-accesslaag naar de domeinlaag gemapped.

Algoritme

Zoals beschreven in het hoofdstuk activity diagram en klassendiagram, heb ik een algoritme ontworpen dat de wensen matchen op verschillende manieren van een productiebedrijf en een personeelslid. Hierbij had ik ook kunnen kiezen voor een andere aanpak door het toevoegen van inherentie en abstractie aan de applicatie. Dit zou hebben geleid tot een betere implementatie van loose coupling, echter zou dit ook betekenen dat er veel statische hulpmiddelen gebruikt zouden moeten worden om bepaalde functionaliteiten te bereiken, zoals de IGetWeeklyNeedLoop (Bron: https://www.c-sharpcorner.com/UploadFile/yusufkaratoprak/difference-between-loose-coupling-and-tight-coupling/).