**EINDONTWERP**

|  |  |
| --- | --- |
| **Naam** | Siddhart Ghogli |
| **Studentnummer** | 21067392 |
| **Klas** | 1 |

**OPDRACHT**

Geef hieronder een korte beschrijving van je opdracht (een samenvatting van je productvisie) zodat je docent OPT3 ook begrijpt met welke opdracht jij bezig bent (neem dit onderdeel over uit je laatst ingeleverde opdracht en vul aan of wijzig de casus aan de hand van eigen bevindingen of op basis van feedback die je hebt gekregen van docenten en/of medestudenten):

De applicatie is een managementtool waarmee medewerkers projecten kunnen aanmaken, beheren en toewijzen aan medewerkers. Het biedt ook functionaliteit voor het bijhouden van budgetten, klanten en medewerkers. Het doel is om het projectbeheerproces te stroomlijnen en te vereenvoudigen voor bedrijven.

**USER STORIES**

Bij de oplevering van het ontwerp hieronder ben ik uitgegaan van de volgende User Stories (gebruik de ‘+’ rechts van de tabel om een User Story toe te voegen):

|  |
| --- |
| Als manager wil ik klantgegevens kunnen toevoegen en bijwerken zodat ik een overzicht heb van alle klanten van het bedrijf. |
| Als manager wil ik projectinformatie kunnen bekijken en bewerken zodat ik op de hoogte blijf van de voortgang en het budget van elk project. |
| Als manager wil ik medewerkers kunnen toevoegen en beheren zodat ik hun contactgegevens en functies binnen het bedrijf van bijhouden. |
| Als medewerker wil ik projecten kunnen aanmaken en bewerken zodat ik de voortgang en details van mijn toegewezen projecten kan bijhouden. |
| Als medewerker wil ik informatie kunnen bekijken over klanten zodat ik op de hoogte ben van hun contactgegevens en projectgeschiedens. |
| Als medewerker wil ik mijn eigen tijdsregistratie kunnen bijhouden per project zodat ik mijn gewerkte uren nauwkeurig kan vastleggen. |

**EINDONTWERP**

In het eindontwerp heb je Patterns en SOLID-principes toegepast en je hebt Classes toegevoegd die je nodig hebt om je applicatie te laten werken (zoals voor het tonen van een GUI of Command Line voor bijvoorbeeld een menu). In het ontwerp heb je ook de types van je properties, parameters en methodes toegevoegd. Zie het volgende document voor voorbeelduitwerking en aanwijzingen:

‘[OPT3 16.2 - Ontwerp en realisatie met patterns - Toelichting.pdf](https://brightspace.hhs.nl/content/enforced/18624-H-SE-S2OPT3-20_2022_VT/documenten/OPT3%2016.2%20-%20Ontwerp%20en%20realisatie%20met%20patterns%20-%20Toelichting.pdf)’

Afbeelding met tekst, diagram, Plan, Technische tekening

Automatisch gegenereerde beschrijving

\* Controleer of het plaatje goed leesbaar is.

**PATTERNS**

Beschrijf hieronder hoe je het Template Method Pattern en een tweede Pattern hebt toegepast in bovenstaand eindontwerp:

|  |  |
| --- | --- |
| **Pattern** | **Beschrijving** |
| Template Method Pattern | Na(a)m(en) van template method (s): start() in AbstractManagementTool  Het Template Method pattern wordt gebruikt in de AbstractManagementTool class om een standaard structuur te bieden voor verschillende functionaliteiten, terwijl de specifieke implementaties aan de subklassen worden overgelaten. De start() methode zorgt voor de uitvoering van de stappen, terwijl abstracte methoden de functionaliteit definiëren. Dit heb ik gedaan, omdat het process voor het toevoegen en verwijderen van klanten, projecten en medewerkers hetzelfde verloopt. |
| Observer Pattern | Het Observer Pattern wordt gebruikt om wijzigingen in de UrenDeclaratie te observeren en hierop te reageren. Na het wijzigen van de uren wordt het resterende bedrag van het budget berekend en wordt de update methode van de observers aangeroepen, om zo een berichtje te sturen in de terminal. |

**SOLID-PRINCIPES**

Beschrijf hieronder voor twee SOLID-principes hoe je die hebt toegepast in Bovenstaand eindontwerp:

|  |  |
| --- | --- |
| **SOLID-principe** | **Beschrijving (gebruik een spatie als je geen toelichting geeft)** |
| Single Responsibility | Elke klasse heeft 1 verantwoordelijkheid. Alle functionaliteiten van de applicatie zijn gegroepeerd met elkaar en hier heb ik er een klasse van gemaakt. |
| Open-Closed | De klasse zijn open voor uitbreiding, maar gesloten voor wijzigingen. Ik maak gebruik van interfaces en abstract classes om dit te bereiken. Voor het ManagementTool gebruik ik bijvoorbeeld een abstract class en voor de observer een interface |
| Liskov | Dit heb toegepast bij bij de abstracte ManagementTools class. De sub-classes die hiervan gebruik maken moeten allemaal de methodes bevatten die in de abstract class staan. Door gebruik te maken van deze abstracte klasse verandert het verwachte gedrag niet van de classes. |
| Interface Segragation | Ik maak voor de AbstractManagementTool gebruik van meerdere interfaces. Deze interfaces bevatten methodes die met elkaar te maken hebben, Ik heb 2 verschillende interfaces: voor het uitprinten en wijzigen van gegevens. Door de functionaliteiten te splitsen in meerdere interfaces kan ik er voor zorgen dat deze classes alleen methodes gebruikt die nodig zijn. |
| Dependency-Inversion | Om dependency inversion toe te passen heb ik abstracte klassen gemaakt. Dit heb ik bijvoorbeeld gedaan voor de ManagementTool. Dit is een hogere-level module die afhankelijk is van 2 verschillende interfaces. |