**EINDONTWERP**

|  |  |
| --- | --- |
| **Naam** | Anthonyrich Numbi |
| **Studentnummer** | 21090475 |
| **Klas** | 7 |

**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 opdracht dat ik gekozen heb is een casus die bedacht is door school. Het gaat over de COA. COA moet het proces om vluchtelingen behandelen. Ze hebben daarvoor een software nodig waarin ze al deze processen kan handelen. De processen moeten het volgende bevatten:

* In het systeem wordt onderscheid gemaakt tussen beheerders, COA-medewerkers en asielzoekers.
* Een beheerder moet landen, gemeentes en AZC’s kunnen beheren (toevoegen, aanpassen en verwijderen)
* Een beheerder moet een rapportage op kunnen vragen voor de uitkering van vergoedingen aan gemeentes die in het kader van de spreidingswet extra vluchtelingen opvangen.
* Een COA-medewerker moet acties kunnen uitvoeren voor het registreren van vluchtelingen en voor de afhandeling van hun asielaanvraag.
* Een vluchteling moet de gegevens op kunnen vragen die over hem zijn geregistreerd.
* Een vluchteling moet een verhuizing kunnen registreren als hij/zij/hen een woning toegewezen heeft gekregen.

**USER STORIES**

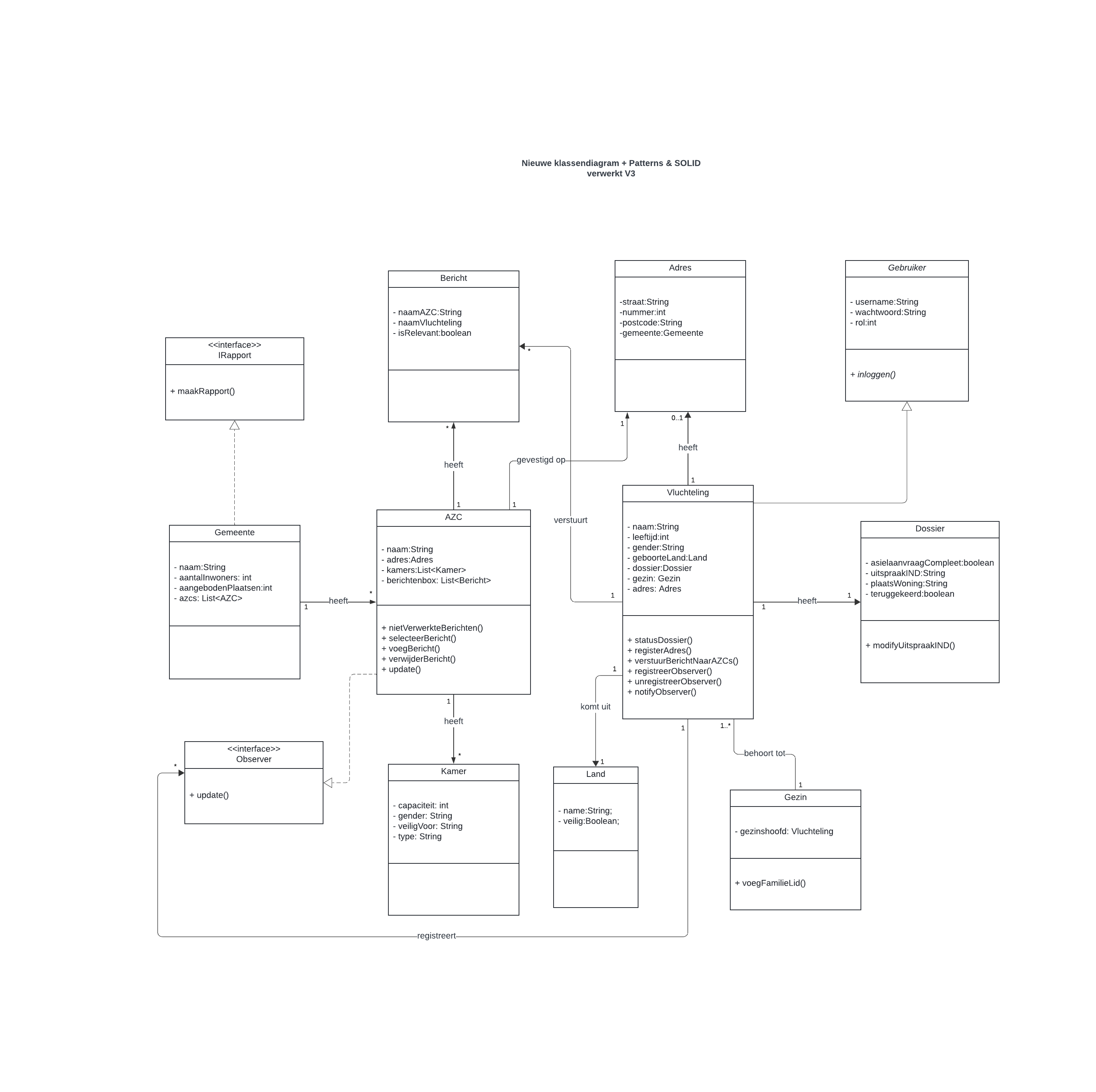
Bij de oplevering van het ontwerp hieronder ben ik uitgegaan van de volgende User Stories (gebruik de ‘+’ rechts van de tabel om een User Story toe te voegen):

|  |
| --- |
| Als **beheerder** wil ik een nieuw land toe kunnen voegen, zodat COA-medewerkers kunnen vastleggen uit welk land van herkomst zij zijn gevlucht. |
| Als **beheerder** wil ik van een bestaand land kunnen wijzigen of het land veilig is of niet, zodat daarmee bij de afhandeling van de asielaanvraag rekening gehouden kan worden. |
| Als **beheerder** wil ik gemeentes kunnen toevoegen, zodat voor AZC’s en het definitieve adres van een statushouder (een asielzoeker die een tijdelijke of definitieve verblijfsvergunning heeft gekregen) kan worden vastgelegd in welke gemeente ze gevestigd zijn. |
| Als **beheerder** wil ik AZC’s kunnen beheren (kunnen toevoegen, kunnen wijzigen of kunnen verwijderen), zodat COA-medewerkers asielzoekers kunnen plaatsen in een beschikbare AZC. |
| Als **beheerder** wil ik een rapportage op kunnen vragen van de uitkeringen die aan gemeentes betaald moeten worden, zodat ik de betalingen aan die gemeentes voor kan bereiden. |
| Als **beheerder** wil ik van een gemeente vast kunnen leggen hoeveel plaatsen er worden aangeboden, zodat een COA-medewerker als eerste uit deze plaatsen kan kiezen voor plaatsing van een nieuwe asielzoeker. |
| Als **COA-medewerker** wil ik een vluchteling kunnen registreren, zodat deze vluchteling in het systeem teruggevonden kan worden. |
| Als **COA-medewerker** kan ik een gemeente opvragen waar ik een asielzoeker in een AZC kan plaatsen (dat kan ook een overplaatsing zijn), zodat ik de asielzoeker kan laten verhuizen. |
| Als **COA-medewerker** wil ik een dossier bij kunnen werken, zodat ik de voortgang van de aanvraagprocedure kan volgen. |
| Als **vluchteling** kan ik de van mij geregistreerde gegevens opvragen, zodat ik eventuele onjuistheden kan bespreken met een COA-medewerker. |
| Als **vluchteling** kan ik de status van mijn dossier opvragen, zodat ik kan inzien hoe de voortgang van mijn asielaanvraag verloopt. |
| Als **vluchteling** wil ik mijn nieuwe adres kunnen registreren, zodat mijn aanvraag kan worden afgerond. |

**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)’



Hier is ook een link naar de Lucidchart file: <https://lucid.app/lucidchart/6c2ed75c-1e49-4ae9-88ca-767417be5e91/edit?viewport_loc=3099%2C-3078%2C5965%2C2835%2CHWEp-vi-RSFO&invitationId=inv_be95474f-5fe9-470a-82e5-6145f47f77c7> (V3) + [PDF(Link)](AsielzoekerOPT.pdf)

\* 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 methode(s): Inloggen()  De Template Method Pattern wordt toegepast door de klassen rondom Gebruikers. Elke type gebruiker krijgt een eigen dashboard te zien. In Gebruiker wordt de structuur van de inlog-methode gemaakt. In de klasse dat Gebruiker implementeren, wordt deze inlog-methode verder volledig uitgewerkt die past bij de logica dat er wordt vereist van de methode. Dit zorgt ervoor dat code steeds niet herhaal wordt, doordat het al in de klasse Gebruiker staat, en ook de code meer overzichtelijker maakt. |
| Observer Pattern | De Observer Pattern wordt toegepast door de klassen Dossier, Vluchteling en AZC. De klasse Vluchteling wordt als subject gebruikt om de observers aan te maken en ze ook weer te verwijderen. Ook wordt de klasse gebruikt door middel van de methode notifyObservers(), om de observers te notificeren wanneer er aanpassingen in een dossier gebeurt. De notificatie wordt naar de AZC's' verstuurd, zodat zij ook op de hoogte zijn van alle aanpassingen rondom de vluchtelingen. |

**SOLID-PRINCIPES**

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

|  |  |
| --- | --- |
| **SOLID-principe** | **Beschrijving (gebruik een spatie als je geen toelichting geeft)** |
| Single Responsibility | Het Single Responsibility Principle wordt toegepast via de Interface IRapport. In IRapport wordt er alleen maar een concept gemaakt voor het genereren van een rapportage. Hierdoor hoeft de klasse Beheerder daar niet op te focussen, door de rapport Functionaliteiten gescheiden te houden. De Rapporten kunnen hierdoor ook makkelijker worden aan gepast en worden uitgebreid in de klassen waar ze worden geïmplementeerd. |
| Open-Closed | Gebruiker klasse is een voorbeeld van een Open/Closed Principle. Open voor uitbreiding, maar gesloten voor wijzigingen. Doordat Gebruiker een child-klasse alles kan laten overerven, kunnen er uitbreidingen plaatsvinden (Bijvoorbeeld als er een nieuwe type gebruiker komt en deze te laten erven van de klasse Gebruiker). Niks hoeft gewijzigd te worden in Gebruiker, waardoor de klasse gesloten staat voor wijzigingen.  Een ander voorbeeld is de klasse Kamer. Momenteel kunnen zijn er 3 type kamers, maar er kan later een nieuwe type kamers worden toegevoegd(Open). Doordat alles al staat in de Kamer klasse, hoeft er niks in de klasse veranderd te worden (Closed). |
| Liskov | LSP wordt toegepast op de gebruiker klasse. Hoe het hier gebruikt wordt doordat COAMedewerker, Beheerder, AZCMedewerker en Vluchteling child-klasses zijn van Gebruiker, kunnen deze klasse als vervangende klassen voor Gebruiker gebruikt worden. Wanneer deze klassen gebruikt worden, verandert er niets aan het gedrag van het programma. Met deze toepassing kunnen we er ook voor zorgen dat de programma verschillende inlog logica heeft voor iedere gebruikerstype, waardoor iedere gebruiker een eigen dashboard kan krijgen dat relevant voor hen is, terwijl het richtlijnen volgt van de klasse Gebruiker. |
| Interface Segragation | Doordat er met verschillende dashboards gewerkt gaat worden, kunnen we hiervoor de Interface Segregation Principle toepassen. Voor elke gebruikerstype wordt er een dashboard gemaakt waarop informatie staat dat relevant voor hen is. Omdat het in de form van een interface is, kunnen we elke klasse methods meegegeven die relevant voor hen is. Dit zorgt ervoor dat interfaces niet geïmplementeerd worden in klasse waar ze niet nodig zijn. |
| Dependency-Inversion |  |