Ontwerpdocument

**Iteratie:** 4

**Groep:** Ontdek Station

Inhoud

[Database Diagram 3](#_Toc187139800)

[C2 Diagram 4](#_Toc187139801)

[C3 Diagram 5](#_Toc187139802)

[Class diagram 6](#_Toc187139803)

# Database Diagram

Afbeelding met tekst, diagram, Plan, Technische tekening

Automatisch gegenereerde beschrijving

# C2 Diagram

Afbeelding met tekst, schermopname, Lettertype, diagram

Automatisch gegenereerde beschrijving

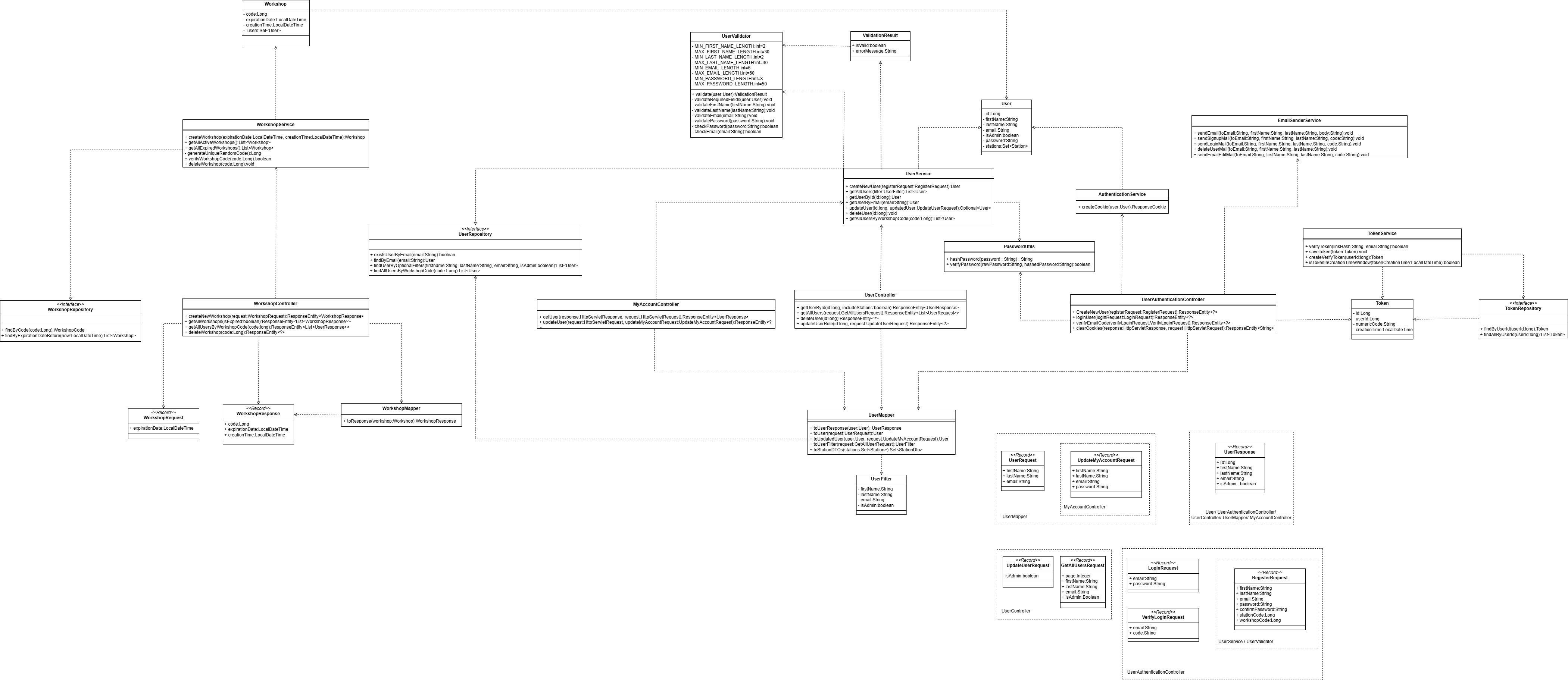
# C3 Diagram

Afbeelding met tekst, schermopname, diagram, Lettertype

Automatisch gegenereerde beschrijving

# Class diagram

Open in de desktopversie van Word om het goed te kunnen lezen.



**Request/Responses:**

* **Containers**: Rechts onder in de klassendiagram vind je een aantal blokken met klassen erin, dit zijn de containers, ze worden gebruikt als vervanging van de pijlen, net als bij de pijlen zijn ze gestippeld, omdat het een implementatie is. Links onderin elke container vind je een naam, deze naam is bij welke klasse de requests/responses behoren. Bijv. UserMapper, MyAccountController, etc.
* **Request**: Dit is wat nodig is van wat de user heeft ingevuld, een request is ook wel bekend als een DTO.
* **Response**: Dit is wat de user terugkrijgt van de computer nadat er een request gestuurd is, een response is ook weer een DTO.
* **<<Record>>**: een record is een soort class, maar aan een record kun je later de variabelen niet meer aanpassen waardoor het betrouwbaarder is.

**Relations:**

* **Gestippelde lijn met open pijl:**  Dit geeft aan dat de relatie tussen de classes een implementatie is, dit betekent dat de ene class de andere gebruikt, welke class de andere gebruikt is te zien aan de pijl, de class waar de pijl naartoe wijst wordt gebruikt door de class waar de pijl vandaan komt, bijv. WorkshopController gebruikt zowel, WorkshopService als WorkshopRequest
* **Containers:** De containers hebben een stippellijn om zich heen wat wijst op een implementatie, de requests/responses dat erin zitten worden gebruikt op dezelfde manier als links de WorkshopRequest wordt gebruikt door de WorkshopController. Alleen dan wordt het gebruikt door de class die linksonder in de container staat.

**Class typen:**

* **<<Interface>>:** In Java door gebruik te maken van JpaRepository in je code is een interface de plek waar je queries gebruikt/maakt. Bijv. UserRepository waar alle methodes staan die gebruik maken van een query.
* **Hoofd-classes:**  De hoofd-classes zijn de classes die vaak maar 1 woord in de classname hebben, bijv. User, Token, etc. Of alleen maar attributen, bijv. ValidationResult. Dit zijn vaak je entiteiten.
* **Service-classes:** In Java zijn de Service-classes hetzelfde als de BLL laag zou zijn in C#, hierin wordt dus alle logica gedaan, bijv. createNewUser, createWorkshop, etc.
* **Controller-classes:** Er wordt gebruik gemaakt van een Client-server architectuur, dit betekent dat er een aparte front- en back-end is, er zijn dus classes nodig die de brug hiertussen mogelijk maakt, hiervoor dienen de controllerclasses. Deze classes ontvangen de verzoeken van de gebruikers, roepen de Serviceclasses aan en sturen de resultaten terug.
* **Aparte classes:** Als laatste hebben we nog de aparte classes, dit zijn de classes die eigenlijk bij geen van de bovengenoemde structuren hoort. Voorbeelden hiervan zijn:

1. **Mappers:** mapperclasses dienen om gegevens over te zetten van het ene type naar het andere, bijvoorbeeld van een entiteit naar een request, dit wordt gebruikt zodat je niet direct de entiteit aan hoef te roepen waar alles in staat, maar in plaats daarvan kunt kiezen om een request(Dto) te gebruiken waar alleen de benodigdheden in staan.
2. **Validator:**  de validator class wordt gebruikt om alle gegevens te controleren. Bijv. UserValidator met de methode validatePassword()
3. **Utility:** utility-classes zijn er om herbruikbare methodes bij elkaar te verzamelen waardoor ze niet in verschillende logica classes opnieuw aangemaakt hoeven te worden.

**Class taken:**

* **User/Account classes:** De User classes doen alles beheren rondom de gebruiker, bijv. registreren, inloggen, gegevens wijzigen, etc.
* **Token classes:** Token classes horen bij het inloggen/registreren, ze creëren, verifiëren en slaan de codes op die meegestuurd worden met de mail, deze codes zijn nodig om je emailadres te bevestigen.
* **Mail class:** Deze class beheert alles rondom het versturen van de mails, voor bijv. het registreren of verwijderen van een user.
* **Workshop classes:** De workshop classes doen alles rondom het aanmaken, oproepen of verwijderen van workshops.