**Vývojový deník – projekt č. 12**

**Říjen**

11.10. – Domluvili jsme se na harmonogramu prací a vytvořili sdílený repositář na GitHubu -> [ChaloupkaM01 · GitHub](https://github.com/ChaloupkaM01/_uois/tree/v2.1)

13.10. – **Konzultace přes MS Teams**   
Vyřešili jsme problém se spuštěním docker-compose.yml. Ujasnili si zadání k projektovému dni (vytvoření prosté ukázky pomocí Apollo a gql\_events, pochopení práce s Apollo). Jak už jsme znali s přechozích semestrů, pochopit zadání „zákazníka“ je jednou z nejdůležitějších částí projektu.

17.10. – Dokončili jsme ukázky k projektovému dni, která spočívala ve vytvoření jednoduché datové struktury „Project“ pomocí Apolla.

18.10. – **Projektový den**

25.10. – **Konzultace přes MS Teams**  
Ujasnili jsme si struktury jednotlivých modelů a začali navrhovat SQL alchemy modely v JupyterNotebooku. Pro zjednodušení jsme využili Jupyter notebooků vytvořených v předcházejícím semestru, kde jsme se učili práci s SQL alchemy a postgres databází.

**Listopad**

2.11. – Povedlo se nám spojit JupyterNotebook s datábazí a SQL modely propsat do PGadminu.

4.11. – Dodělali jsme SQL alchemy modely v JupyterNotebooku a připravili struktury testovacích dat pro vytvoření plnící funkce do databáze tzv. feederu. Na konzultaci jsme si ujasnili další postup. Udělat importní funkci na zpracování struktury poté udělat strawberry GQL modely a napsat pro ně resolvery. Inspirovat se projektem gql\_ug.

15.11. – SQL alchemy modely navrhnuté v JupyterNotebooku fungují, a tak jsme je přepsali do dbDefinitions a úspěšně propojili s databází.

22.11. – Marný pokus o začátek dbFeederu, kvůli celkovému nepochopení jeho role v projektu. Nadefinování několika funkcí. Musíme pochopit spojitost feederu a Apolla, feederu a foreign key.

25.11. – **Konzultace přes MS Teams**  
Zkontrolování dbDefinitions. Po hlubším pochopení, jak přesně má pracovat feeder se můžeme pustit do další etapy našeho projektu.

29.11. – **Projektový den**

**Leden**

9.1. – Doladili jsme snad už poslední detaily v dbFeederu a dbDefinitions, vytvořili jsme GraphResolvers a GraphTypeDefinitions.  
Pravděpodobně jsme rozbili Apollo, protože se přestal načítat jeho kontejner, ale projekt funguje i nadále přes GQL endpoint na námi definovaném portu.

10.1. – **Konzultace přes MS Teams**   
Projekt přestal fungovat pravděpodobně kvůli staré verzi projektu. Dodělali jsme drobné nedostatky v téměř v celém projektu.

11.1. – Podařilo se nám zprovoznit feeder. Pomocí natvrdo vytvořené skupiny s předem zadaným UUID úspěšně naplníme data do databáze. Na první pohled vše funguje, jak by mělo. Tabulky mají správné atributy i relationships.

12.1. – **Projektový den**  
Před dalším vývojem musíme aktualizovat verzi našeho projektu na nejnovější verzi celkového projektu. Touto „aktualizací“ bychom měli opravit nefungující Apollo.

**Únor**

9.2. – Pokus o implementaci do stávající verze funkční verze UOIS. Pokus byl neúspěšný z důvodu nejasné úpravy našeho kódu a stále nefunkčního Apolla.

13.2 – Znovu pokus o implementaci do UOIS. Opět neúspěšný. Vzdali jsme to a rozhodli se pokračovat v “lokální/oddělené“ verzi. Drobné úpravy GraphTypeDefinitions dle potřeb (validace, změna u datumu z DateTime na Date)

14.2 – Začátek vytváření editoru pro projekty. Řešení zapeklitého problému, proč se editor nezobrazuje v GQL endpointu. Řešení bylo prosté => sice jsme nadefinovali ProjectEditorGQLModel, ale v ProjectGQLModel nebyla editor nadefinován.

15.2 – Dokončení editoru, finální testování, dokončení dokumentace, vytvoření docker image, final push

**LESSONS LEARNED**

-v praxi otestována teoretická znalost relací v databázi 1:1, 1:N

-první cesta není vždy ta správná

-stupňovat složitost řešených problémů (komplexní problémy dělit na menší části)

-v budoucích projektech se více zamyslet nad time managementem

-když si nevíš rady, obrať se pro pomoc!