



ABSTRAKT

Dokumentácia k práci na
predmetoch tímového vývoja
softvéru na FMFI UK v
akademickom roku
2024/2025

VISCINDY

Dokumentácia k riadeniu projektu

Členovia tímu:

Jakub Vojtek
Matej Budoš
Norbert Hašan
Adrián Kýška
Jozef Kolek
(Marek Danihel)

Role:

Documentation Leader
Python developer
Unity developer
Python developer
Unity developer

Cvičiaci: Juraj Vincúr

GitHub:

<https://github.com/MatejBudos/VisCindy>

I. šprint (ZS)

Začiatok šprintu: 8.11. 2024

Koniec šprintu: 22. 11. 2024

Stories

Názov	Typ	Story points	Riešitelia	Stav akceptácie
Prvotný prototype v Unity	Infrastructure	5	Norbert Hašan , Jozef Kolek, Jakub Vojtek	Prijatá
Prvotný prototype služby v Pythone	Infrastructure	8	Matej Budoš , Marek Danihel, Adrián Kýška	Zamietnutá
Iniciálny návrh	Management	4	Jakub Vojtek	Prijatá

Retrospektíva

Problém	Action item(s)
Nikto nemal user stories na zodpovednosti.	Priradovať user story vždy niekomu kto za ňu bude zodpovedať a bude dohliadať na aktuálny stav.
Tasky neboli reviewed žiadnym iným členom	Priradiť niekoho ku každému tasku z danej skupiny kto sa na to pozrie (spraví review).

II. šprint (ZS)

Začiatok šprintu: 22. 11. 2024

Koniec šprintu: 6.12.2024

Stories

Názov	Typ	Story points	Riešitelia	Stav akceptácie
Prvotný prototype služby v Pythone	Infrastructure	8	Matej Budoš, Adrián Kýška	Prijatá
Implementácia object pooling-u	Infrastructure	6	Norbert Hašan, Jozef Kolek, Jakub Vojtek	Prijatá
Vizuálny návrh editácie a priradenia cypher queries úpravám	Management	8	Jakub Vojtek, Matej Budoš	Prijatá
Doplnenie iniciálneho návrhu	Management	8	Jakub Vojtek, Matej Budoš	Prijatá

Retrospektíva

Problém	Action item(s)
Viacerí členovia tímu začali svoje úlohy riešiť deň pred koncom šprintu.	Každý člen tímu rozpracuje každý svoj task pred polovicou šprintu.
Opustil nás jeden člen tímu.	Rozdelíme si jeho časť práce.

III. šprint (ZS)

Začiatok šprintu: 6.12.2024

Koniec šprintu: : 20.12.2024

Stories

Názov	Typ	Story points	Riešitelia	Stav akceptácie
Práca s viacerými grafmi	Infrastructure	8	Matej Budoš, Adrián Kýška	Prijatá
Prototyp filtrovania grafu pomocou cypher queries	Infrastructure	8	Jakub Vojtek, Norbert Hašan	Prijatá
Prototyp editácie grafu + doplnenie návrhu	Management	13	Jozef Kolek, Norbert Hašan, Adrián Kýška, Jakub Vojtek	Prijatá

IV. šprint (ZS)

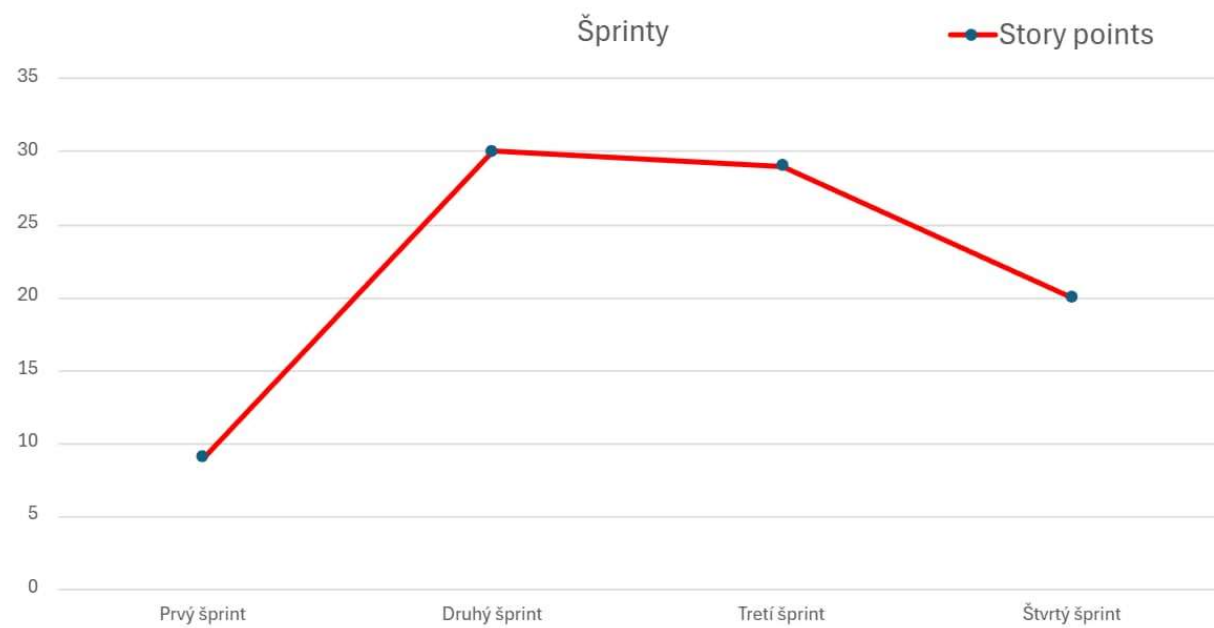
Začiatok šprintu: 3.1.2025

Koniec šprintu: 17.1.2025

Stories

Názov	Typ	Story points	Riešitelia	Stav akceptácie
Visual query a implementácia building query	Infrastructure	8	Norbert Hašan, Jakub Vojtek	Prijatá
Napojenie na backend a zobrazenie dopytu	Infrastructure	6	Matej Budoš, Jozef Kolek,	Prijatá
Dokumentácia	Auxiliary	6	Jakub Vojtek, Adrián Kýška, Jozef Kolek	Prijatá

Grafická reprezentácia story pointov počas priebehu vývoja



Analytická dokumentácia

Úvod

Cieľom projektu je vytvoriť softvér, ktorý umožní užívateľom editovať a filtrovať grafy vo virtuálnej realite. Tento nástroj je určený na vizualizáciu komplexných vzťahov a údajov vo forme grafov, čo uľahčuje ich pochopenie a analýzu. Používatelia budú mať možnosť interagovať s grafmi intuitívnym spôsobom, čo prispieva k efektívnejšiemu rozhodovaniu a kreatívnemu prístupu k riešeniu problémov.

Tento projekt je navrhnutý pre široké spektrum užívateľov, vrátane študentov, vedcov a profesionálov, ktorí pracujú s komplexnými sieťovými údajmi. Naším zámerom je zabezpečiť jednoduché používanie softvéru aj pre užívateľov bez technického zázemia, s dôrazom na prehľadnosť a intuitívne ovládanie.

Hlavné požiadavky

Softvér umožní užívateľom flexibilne formulovať dotazy, ktoré môžu zadávať manuálne alebo vytvárať prostredníctvom vizuálneho nástroja, čím sa odstráni potreba znalosti zložitých jazykov. Výsledky týchto dotazov sa okamžite zobrazia v trojrozmernom priestore, čo umožňuje lepšie pochopenie vzťahov medzi prvkami grafu. Užívatelia budú mať možnosť pridávať do grafu nové vrcholy a hrany na reprezentáciu údajov, upravovať ich alebo ich odstraňovať, a to s podporou funkcie „Krok späť“ na zrušenie nežiaducich zmien.

Na optimálne zobrazenie grafov budú k dispozícii algoritmy automatického rozloženia, ktoré zaručia prehľadnosť a jasnú vizualizáciu vzťahov. Okrem toho si užívatelia budú môcť prispôbiť vzhľad grafov podľa svojich preferencií. Softvér umožní spravovať viaceré grafy naraz, čo zahŕňa možnosť ich uloženia, prepínania medzi nimi a importovania alebo exportovania údajov pre ďalšiu analýzu a zdieľanie.

Integrácia virtuálnej reality poskytne trojrozmerný priestor, v ktorom budú grafy zobrazované. Používatelia budú mať možnosť intuitívne manipulovať s grafmi prostredníctvom gest a pohybov, čo poskytne prirodzený a pohodlný spôsob ovládania. Tento projekt spája inovácie v oblasti vizualizácie a technológií virtuálnej reality s cieľom vytvoriť užitočný a efektívny nástroj pre analýzu grafov a komplexných vzťahov.

Use case diagram

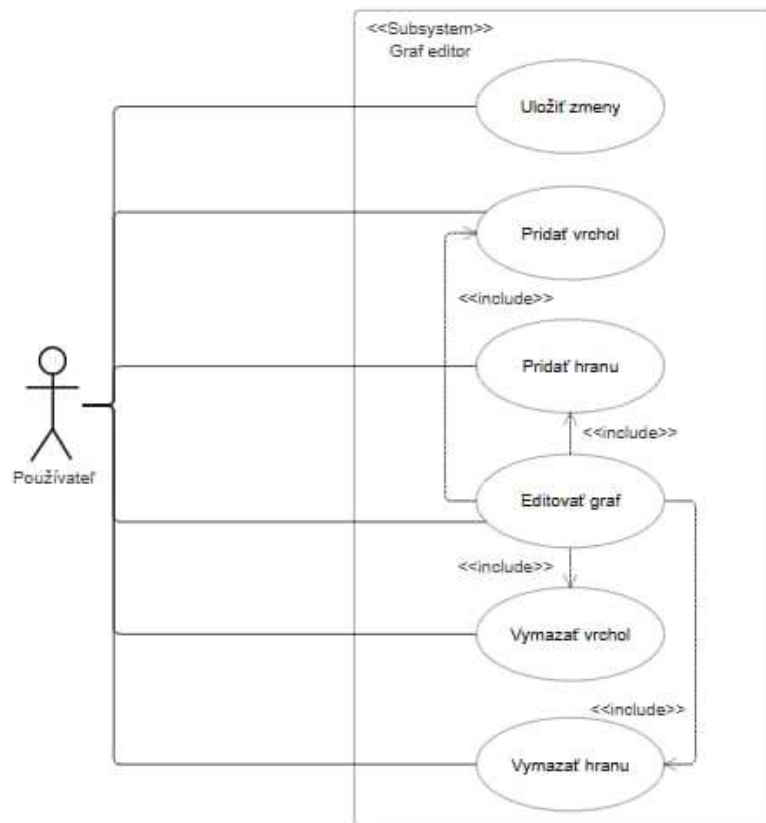


Diagram znázorňuje interakciu používateľa so systémom grafického editora a modeluje jednotlivé funkcionality, ktoré systém poskytuje. Používateľ (aktér) má k dispozícii nasledovné možnosti v rámci systému:

1. Uložiť zmeny: Umožňuje používateľovi uložiť vykonané úpravy grafu.
2. Pridať vrchol: Používateľ môže do grafu pridať nový vrchol (uzol).
3. Pridať hranu: Používateľ môže pridať hranu, ktorá spája dva vrcholy.
4. Editovať graf: Táto funkcionality zahŕňa operácie, ako je pridanie vrcholu, pridanie hrany, vymazanie vrcholu alebo vymazanie hrany.
5. Vymazať vrchol: Používateľ môže odstrániť existujúci vrchol z grafu.
6. Vymazať hranu: Používateľ má možnosť odstrániť hranu spájajúcu dva vrcholy.

Use Case #1: Pridanie nového vrcholu

Level: User-goal

Primary Actor: Používateľ

Predpoklady:

- Používateľ má už otvorený graf.
- Systém (Unity, Flask, Neo4j) je pripojený a funkčný.

Hlavný úspešný scenár:

1. Používateľ prejde do rozhrania na úpravu grafu.
2. Používateľ zadá názov nového vrcholu (napr. "Vrchol 1") do vstupného poľa.
3. Systém (Unity) pridá nový vrchol lokálne do grafu a prekreslí ho.
4. Používateľ môže vykonať ďalšie zmeny v grafe (pridanie hrán, úprava vrcholov).
5. Používateľ skontroluje graf a potvrdí, že je spokojný s vykonanými zmenami.
6. Systém odošle REST API požiadavku na Flask server s údajmi o všetkých zmenách.
7. Flask server zostaví Cypher dotaz pre pridanie vrcholu (a ďalších zmien, ak sú).
8. Flask odošle Cypher dotaz do databázy Neo4j.
9. Databáza uloží nový vrchol a ďalšie zmeny.
10. Flask server odošle spätnú väzbu o úspechu operácie do Unity.
11. Používateľ vidí potvrdenie, že zmeny boli úspešne uložené.

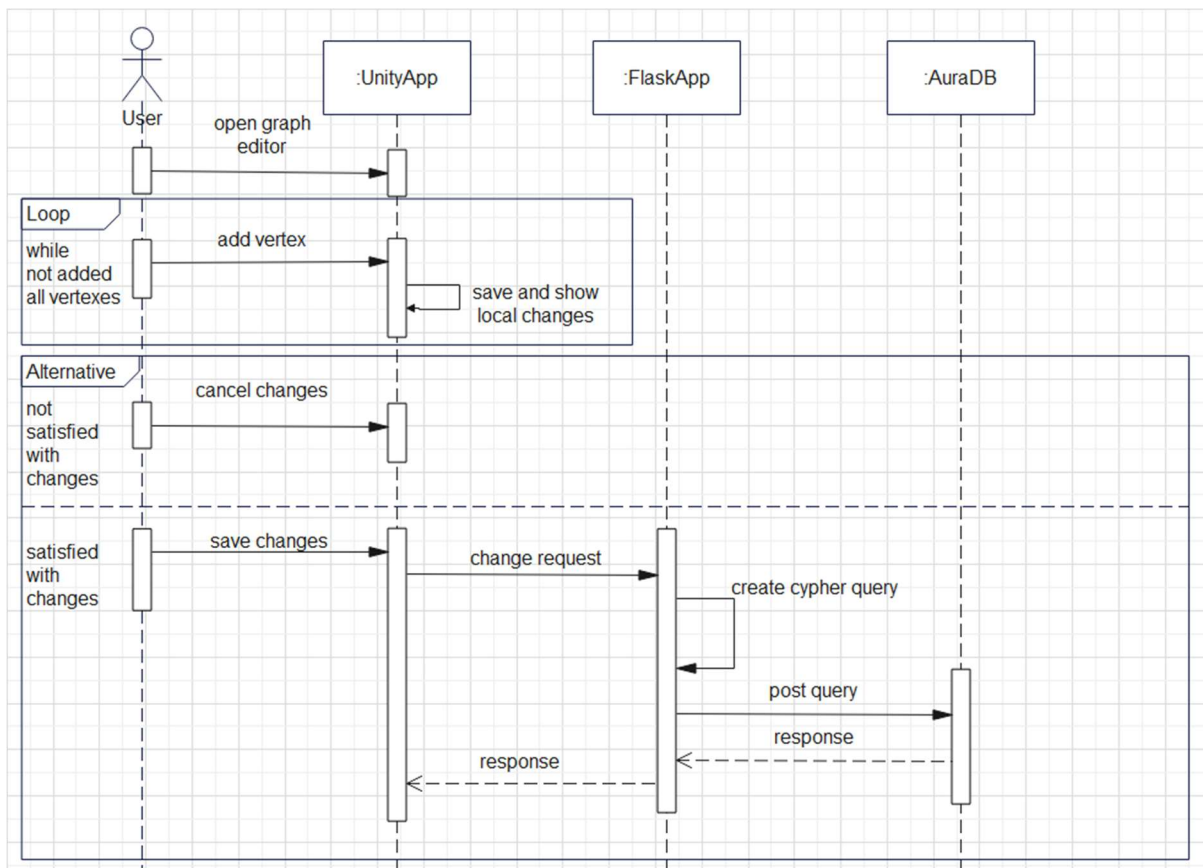
Alternatívne toky:

5a. Používateľ chce zrušiť všetky zmeny:

- Systém vráti graf do pôvodného stavu (pred začiatkom úprav).

Výsledok:

- **Úspech:** Nový vrchol a ďalšie zmeny sú uložené v databáze a zobrazené v Unity.
- **Zlyhanie:** Používateľ je informovaný o chybe a môže zopakovať proces ukladania zmien.



Use Case #2: Editovanie grafu

Level: User-goal

Primary Actor: Používateľ

Predpoklady:

- Používateľ má už otvorený graf.
- Systém (Unity, Flask, Neo4j) je pripojený a funkčný.

Hlavný úspešný scenár:

1. Používateľ vyberie vrchol alebo hranu, ktorú chce upraviť, v Unity aplikácii.
2. Zobrazí sa dialógové okno, kde môže používateľ upraviť atribúty vybraného prvku (napr. zmeniť názov vrcholu alebo aktualizovať váhu hrany) alebo prvok odstrániť.
3. Systém (Unity) aplikuje zmenu lokálne na vizualizáciu grafu.
4. Používateľ vykoná ďalšie úpravy podľa potreby alebo skontroluje aktuálny stav grafu.
5. Keď je používateľ spokojný s úpravami, potvrdí ich odoslanie.
6. Systém odošle REST API požiadavku na Flask server spolu so všetkými upravenými atribútmi.

7. Flask server zostaví Cypher dotaz pre vykonanie zmien v databáze.
8. Flask odošle Cypher dotaz do databázy Neo4j.
9. Databáza vykoná požadované zmeny. ---
10. Flask server odošle spätnú väzbu o úspechu operácie do Unity.
11. Používateľ vidí potvrdenie, že zmeny boli úspešne uložené.

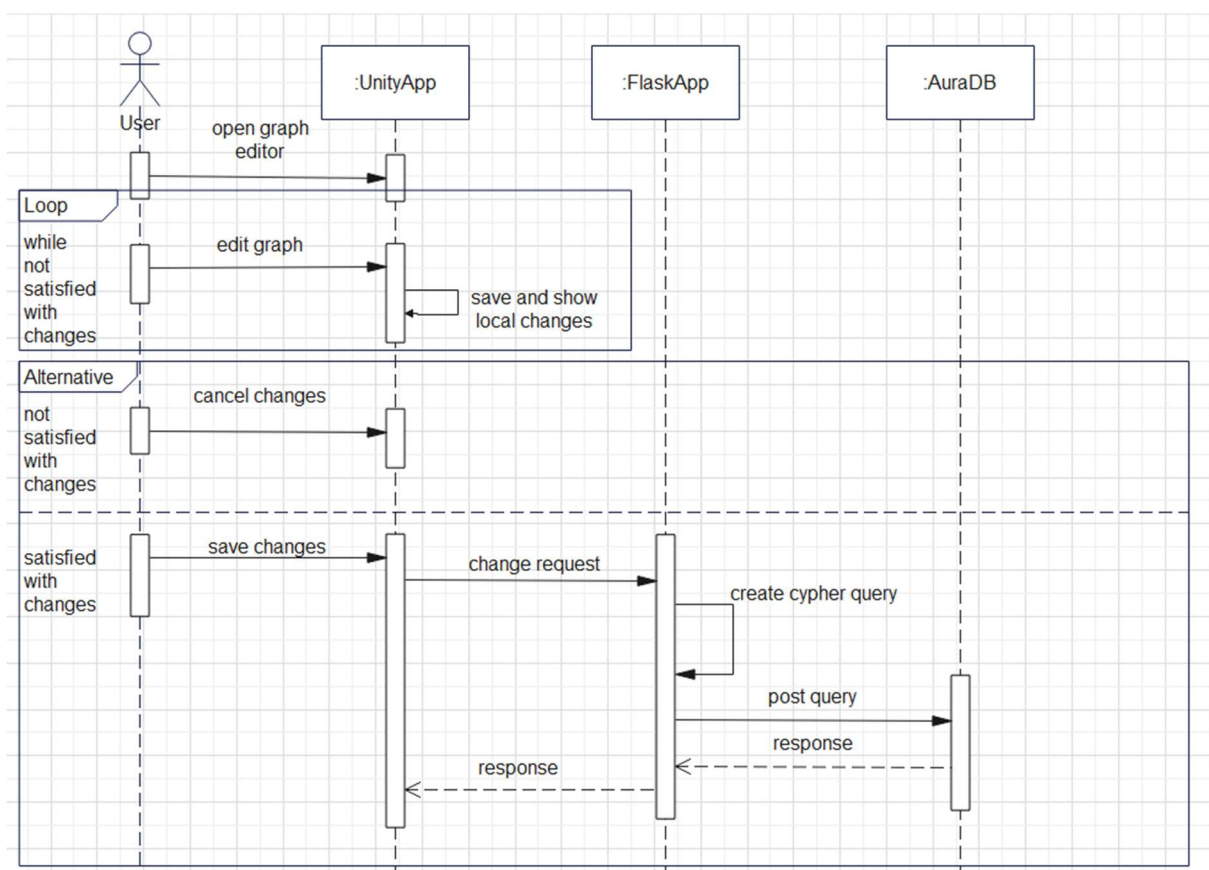
Alternatívne toky:

5a. Používateľ chce zrušiť všetky úpravy:

- Systém vráti graf do pôvodného stavu (pred začiatkom úprav).

Výsledok:

- **Úspech:** Upravený vrchol alebo hrana (alebo ich odstránenie) sú uložené v databáze a zobrazené v Unity.
- **Zlyhanie:** Používateľ je informovaný o chybe a môže zopakovať proces ukladania zmien.



Use Case #3: Zmena layoutu

Level: User-goal

Primary Actor: Používateľ

Predpoklady:

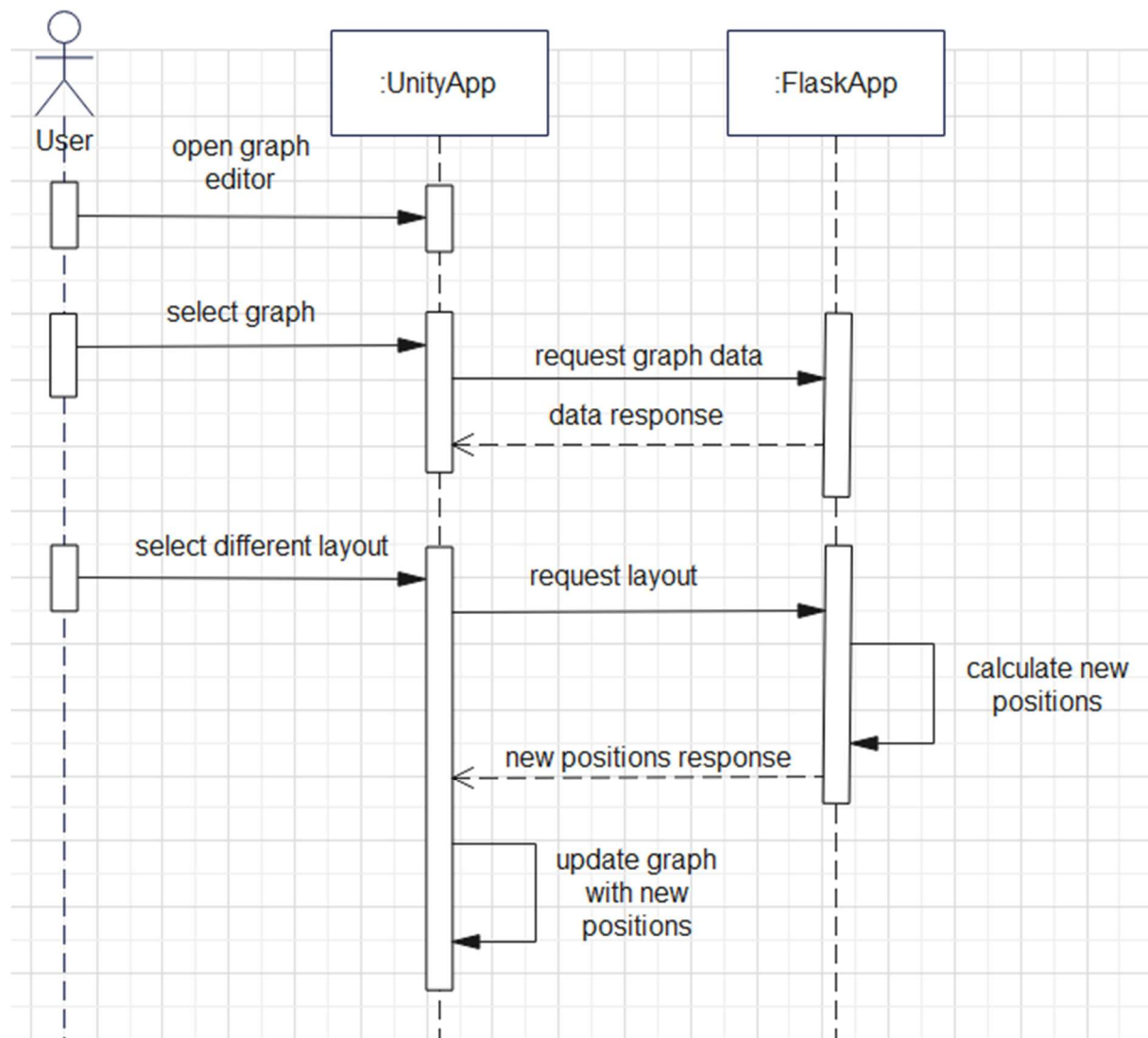
- Používateľ má už otvorený graf.
- Systém (Unity a Flask) je pripojený a funkčný.

Hlavný úspešný scenár:

1. Používateľ otvorí požadovaný graf v Unity aplikácii.
2. Používateľ prejde do nastavení grafu a vyberie požadovaný layoutový algoritmus zo zoznamu.
3. Používateľ potvrdí výber.
4. Unity aplikácia odošle REST API požiadavku na Flask server s informáciou o zvolenom layoutovom algoritme a aktuálnych údajoch grafu.
5. Flask server prijme požiadavku, vykoná výpočet nových pozícií vrcholov podľa zvoleného algoritmu.
6. Flask server odošle nové pozície vrcholov a hrán späť do Unity aplikácie.
7. Unity aplikácia prekreslí graf na základe nových pozícií a zobrazí výsledok používateľovi.
8. Používateľ vidí graf v novom rozložení a môže pokračovať v ďalších úpravách alebo analýze grafu.

Výsledok:

- **Úspech:** Graf je zobrazený s novým rozložením podľa zvoleného layoutového algoritmu.
- **Zlyhanie:** Používateľ je informovaný o chybe a graf zostane v pôvodnom stave.



Technická dokumentácia

Frontend

Frontend riešenia využíva Unity 3D ako hlavný nástroj na zobrazovanie grafov a manipuláciu s nimi v trojrozmernom priestore. Táto technológia umožňuje plynulú vizualizáciu komplexných vzťahov a pridáva podporu pre interakciu vo virtuálnej realite. Na vývoj VR funkcionalít je integrovaný Mixed Reality Toolkit (MRTK), ktorý poskytuje potrebné nástroje na intuitívne ovládanie a gestá. Centralizovaný manažment stavu aplikácie je zabezpečený návrhovým vzorom Singleton, ktorý zaručuje efektívne riadenie všetkých vizuálnych a interaktívnych prvkov aplikácie.

Backend

Backend slúži ako sprostredkovateľ medzi užívateľským rozhraním a databázou. Na jeho implementáciu sa využíva Flask, ktorý poskytuje ľahké a škálovateľné REST API. Toto API umožňuje vykonávať operácie, ako je dotazovanie, úprava a správa grafových dát. Aby bola interakcia s databázou jednoduchá a prehľadná, využíva sa návrhový vzor Facade, ktorý abstrahuje zložité procesy a ponúka zjednodušené rozhranie pre manipuláciu s dátami. Táto architektúra tiež umožňuje rozšíriteľnosť backendových služieb podľa budúcich požiadaviek projektu.

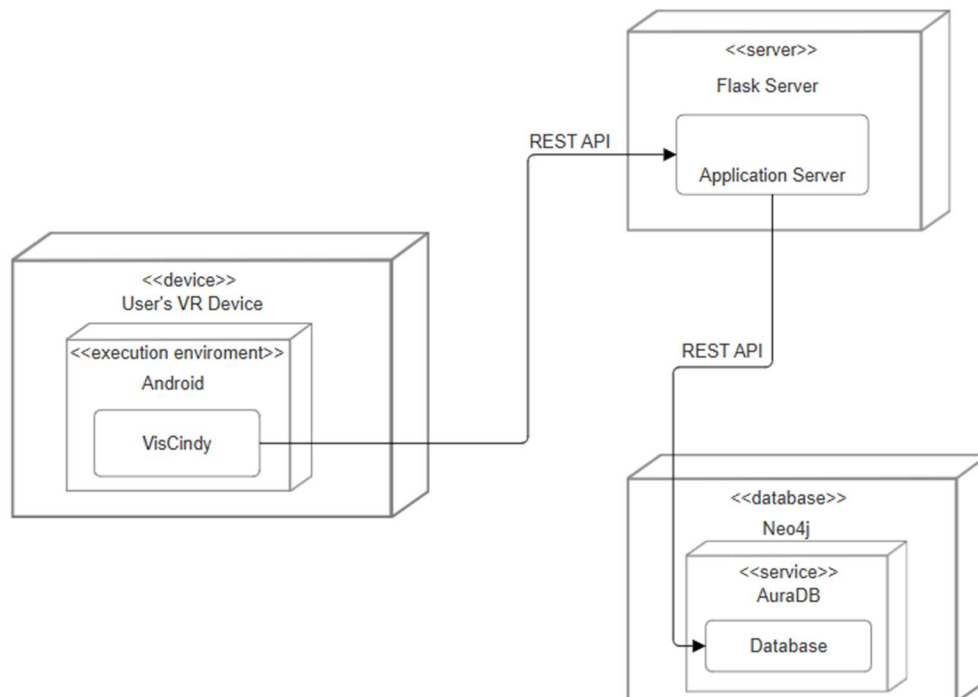
Databáza

Databázová vrstva je postavená na Neo4j, ktorá je optimalizovaná na prácu s grafovými údajmi. Táto technológia umožňuje efektívne dotazovanie pomocou Cypher query language, čo je jazyk špecifický pre grafové databázy. Neo4j zabezpečuje ukladanie, editáciu a načítavanie komplexných vzťahov medzi uzlami a hranami grafu. Okrem toho databáza podporuje import a export dát vo formáte JSON, čím uľahčuje integráciu s inými systémami. Na automatické rozloženie grafov využíva databáza integrované algoritmy, ktoré zlepšujú vizualizáciu a prehľadnosť dát.

Komunikácia

Komunikácia medzi jednotlivými vrstvami aplikácie prebieha prostredníctvom REST API, ktoré spája frontend, backend a databázovú vrstvu. Toto API umožňuje dynamické dotazovanie a operácie s grafmi, ako je pridávanie nových uzlov, úprava existujúcich vzťahov a mazanie prvkov. Na zabezpečenie konzistencie a jednoduchosti práce s dátami je API navrhnuté tak, aby podporovalo transakčný model, ktorý minimalizuje riziko nekonzistencie pri zlyhaní jednotlivých operácií.

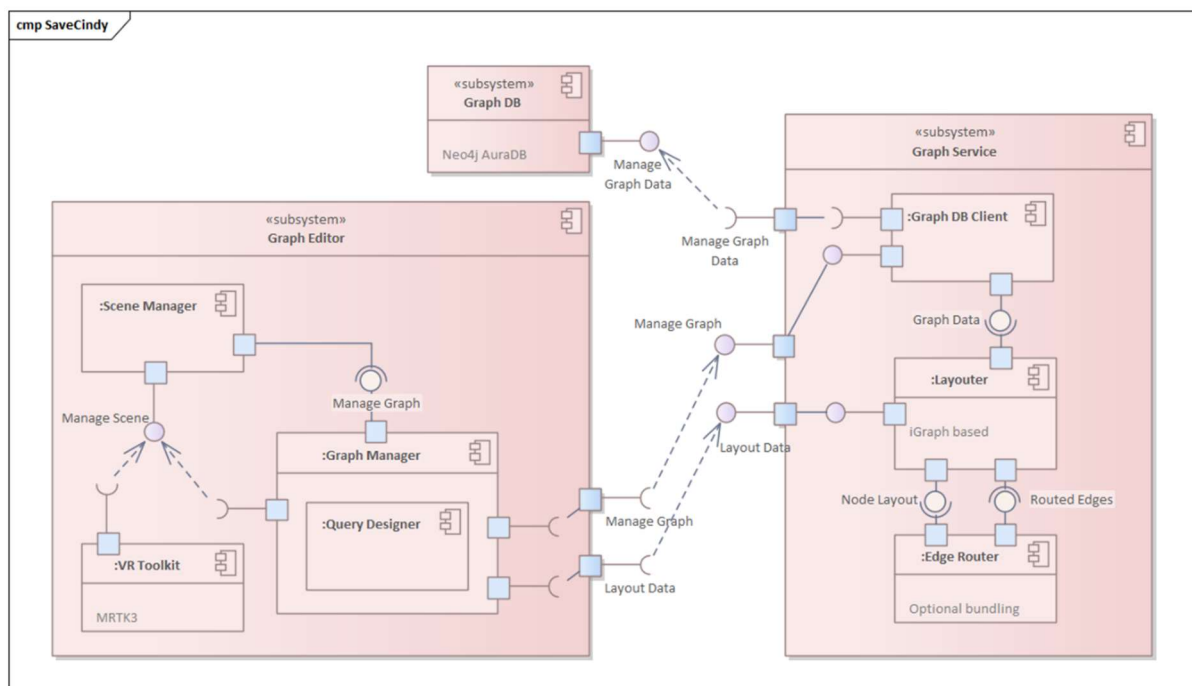
Deployment diagram



User's VR Device zodpovedá za vizualizáciu grafov v 3D priestore a podporuje virtuálnu realitu. Application server poskytuje REST API pre komunikáciu medzi VR device a databázou a abstrahuje prístup k databáze. Databáza je optimalizovaná na ukladanie grafových dát a podporuje efektívne dotazovanie cez Cypher query language.

Komunikácia medzi vrstvami prebieha cez REST API: frontend odosiela požiadavky na manipuláciu s grafovými dátami cez backend, ktorý komunikuje s databázou pre dotazovanie a správu údajov. Komunikácia je podporovaná transakčným modelom, ktorý zabezpečuje konzistenciu dát.

Component diagram



Hlavné komponenty:

1. Graph Editor (Podmodul):

- Zodpovedný za interakciu používateľa s grafom a vizualizáciou. Zahŕňa:
 - **Scene Manager:**
 - Správa scény, ktorá obsahuje objekty a graf.
 - Zodpovedá za načítanie a prechod medzi scénami v aplikácii.
 - **Graph Manager:**
 - Centralizovaný správca pre manipuláciu a správu grafu. Tento komponent koordinuje komunikáciu s backendom aj vykreslením.
 - **Query Designer:**
 - Umožňuje používateľovi tvoriť a vykonávať dotazy nad grafom. Zahŕňa Cypher query builder.
 - **VR Toolkit (MRTK3):**
 - Modul pre podporu VR interakcií, umožňujúci manipuláciu s grafmi vo virtuálnej realite.

2. Graph DB (Podmodul):

- **Neo4j AuraDB:**
 - Úložisko grafových dát. Poskytuje backendové služby na dotazovanie, editáciu a ukladanie grafov.

3. **Graph Service (Podmodul):**

- Abstrakcia pre operácie s grafom. Zahŕňa:
 - **Graph DB Client:**
 - Klient na komunikáciu s databázou Neo4j. Tento komponent zaisťuje spracovanie grafových dát a ich prenos medzi aplikáciou a databázou.
 - **Layouter:**
 - Komponent zodpovedný za vizuálne rozloženie vrcholov v grafe.
 - Využíva rôzne algoritmy na optimalizáciu rozloženia grafov.
 - **Edge Router:**
 - Komponent špecializovaný na routovanie hrán medzi vrcholmi v grafe.
 - Pravdepodobne ponúka funkcie na zlepšenie estetiky vizualizácie, napr. vyhýbanie sa prekryvaniu hrán.

Abstraktný high-level náhľad na biznis objekty

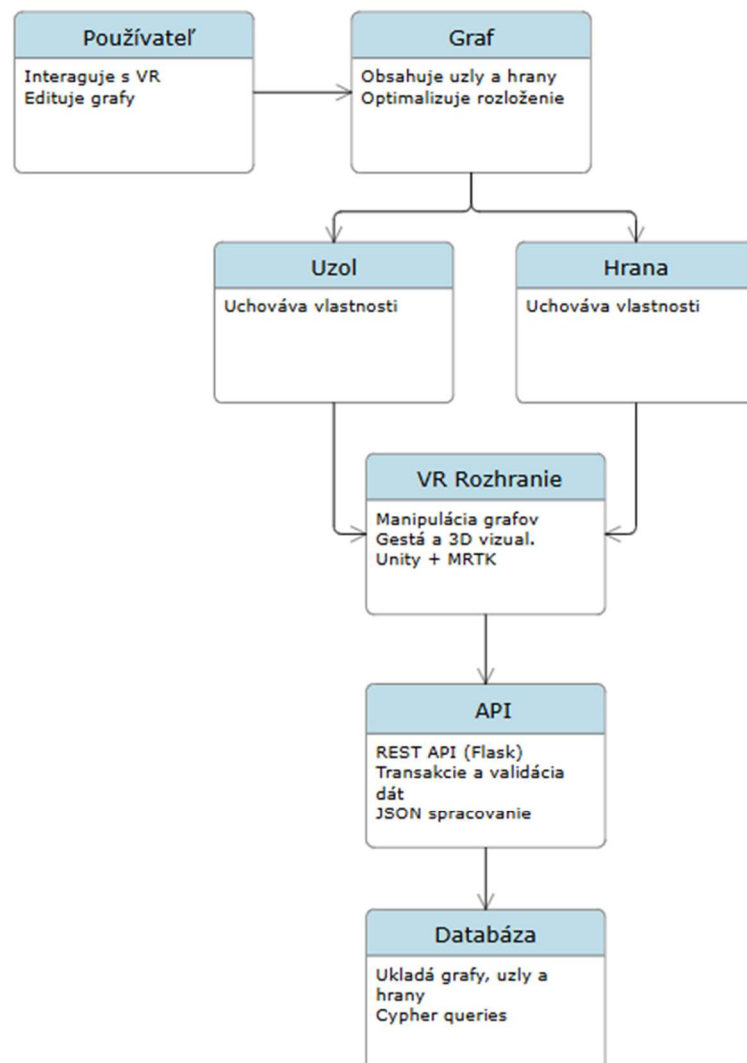


Diagram ukazuje vzťahy medzi jednotlivými biznis objektmi projektu a ich hlavnú funkcionálnosť.

Code snippets

Prepojenie Unity s rest api pre získanie dát grafu.

```
106 private IEnumerator GetGraphData()
107 {
108     using (HttpClient client = new HttpClient(new HttpClientHandler
109     {
110         CookieContainer = CookieContainer,
111         UseCookies = true
112     }))
113     {
114         Debug.Log(getGraphDropdown.options[getGraphDropdown.value].text);
115         var response1:Task<HttpMessage> = client.GetAsync(requestUri.apiUrl + "graph/" +
116             getGraphDropdown.options[getGraphDropdown.value].text);
117         yield return response1;
118
119         if (response1.Result.IsSuccessStatusCode)
120         {
121             _responseData1 = response1.Result.Content.ReadAsStringAsync().Result;
122             yield return _responseData1;
123         }
124         else
125         {
126             Console.WriteLine($"Error: {response1.Result.StatusCode} - {response1.Result.ReasonPhrase}");
127         }
128     }
129 }
130 }
131
```

Vizualizácia grafu a použitie poolu nodov. Object pooling.

```
132 private void VisualizeGraph(Dictionary<string, NodeObject> forAdd)
133 {
134     Queue<GameObject> spherePool = ObjectPool.SharedInstance.poolDictionary[SPHERE_POOL_KEY];
135     Queue<GameObject> linePool = ObjectPool.SharedInstance.poolDictionary[LINE_POOL_KEY];
136
137     foreach (KeyValuePair<string, NodeObject> node in forAdd)
138     {
139         GameObject sphere = spherePool.Dequeue();
140         if (sphere != null)
141         {
142             sphere.SetActive(true);
143             sphere.name = node.Key;
144             sphere.transform.position = new Vector3(node.Value.x, node.Value.y, node.Value.z);
145             sphere.transform.SetParent(graphPrefab.transform);
146             _nodesDictionary[node.Key].UInode = sphere;
147             activeNodes.Add(sphere);
148         }
149         else
150         {
151             Debug.LogWarning(message: "Not enough spheres in the pool!");
152         }
153     }
154
155     foreach (KeyValuePair<string, NodeObject> node in forAdd)
156     {
157         foreach (string targetNode in node.Value.edges)
158         {
159             // ...
160         }
161     }
162 }
```

Implementácia vrátenia sa späť, pri napr posune nodu, alebo vymazaní.

```
public void Redo()
{
    if (redo.Count != 0)
    {
        Command aktualCommand = redo.Pop();
        if (aktualCommand.command.Equals("addNode"))
        {
            SetVisibilityNode(aktualCommand.gameObject, active: true);
            undo.Push(aktualCommand);
        } else if (aktualCommand.command.Equals("addRelationship"))
        {
            SetVisibilityEdge(aktualCommand.gameObject, active: true);
            undo.Push(aktualCommand);
        } else if (aktualCommand.command.Equals("deleteNode"))
        {
            SetVisibilityNode(aktualCommand.gameObject, active: false);
            foreach (KeyValuePair<string, NodeObject> vrchol in _nodesDictionary)
            {
                foreach (KeyValuePair<string, GameObject> edge in _nodesDictionary[vrchol.Key].UIEdges)
                {
                    if (vrchol.Key.Equals(aktualCommand.nodeName) || edge.Key.Equals(aktualCommand.nodeName))
                    {
                        SetVisibilityEdge(edge.Value, active: false);
                    }
                }
            }
            undo.Push(aktualCommand);
        } else if (aktualCommand.command.Equals("deleteRelationship"))
        {
            undo.Push(aktualCommand);
            SetVisibilityEdge(aktualCommand.gameObject, active: false);
        }
    }
}
```

Update dropdown boxu a jeho dát na základe dát s rest api.

```
public void UpdateDropdown()
{
    nodeLabelDropdown.ClearOptions();
    nodePropertiesDropdown.ClearOptions();
    nodeLabelDropdown.AddOptions(_jsonDictionary["node_labels"]?.ToObject<List<string>>());
    nodePropertiesDropdown.AddOptions(_jsonDictionary["node_properties"]?.ToObject<List<string>>());
}
```


Doplnenie zátvoriek aby sedeli na konci. Zachová logický význam.

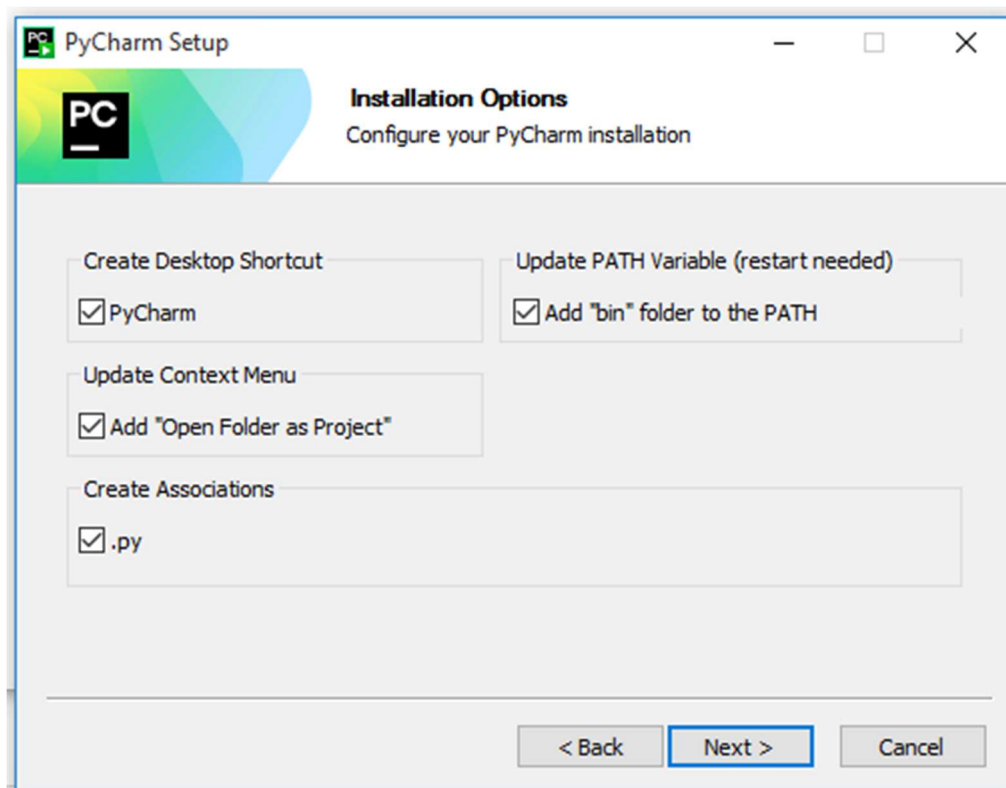
```
Frequently called 2 usages Norbert Haian  
int ParenthesesDifference(string inputString)  
{  
    int count1 = 0;  
    int count2 = 0;  
    foreach (char c in inputString)  
    {  
        if (c == '(')  
        {  
            count1++;  
        }  
  
        if (c == ')')  
        {  
            count2++;  
        }  
    }  
    return count1-count2;  
}
```

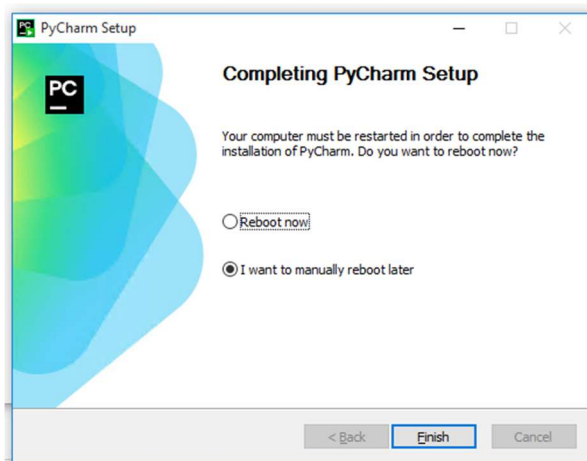
Inštalačná príručka

Najskôr si nainštalujeme pycharm:

<https://www.jetbrains.com/pycharm/download/?section=windows>

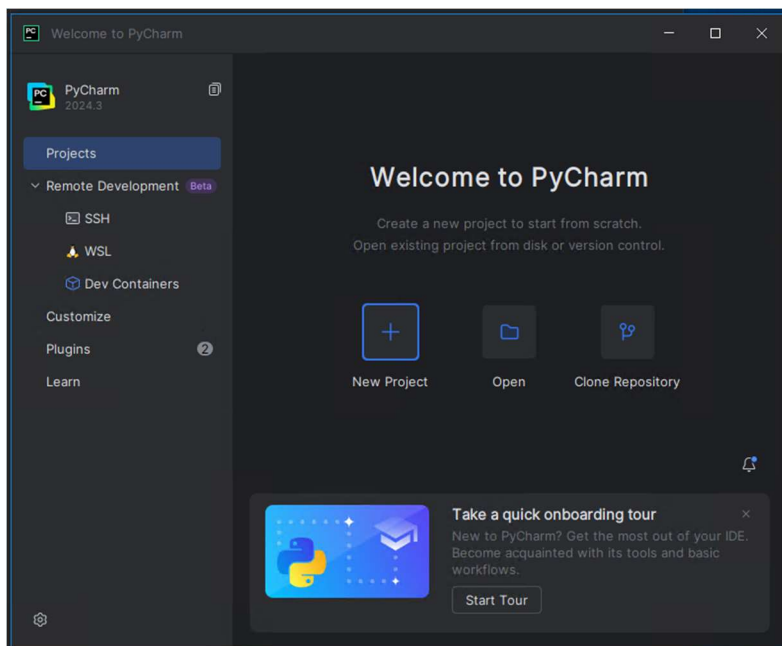
This PC > Local Disk (C:) > Install				
	Name	Date modified	Type	Size
ss	 pycharm-professional-2024.3	28.11.2024 21:53	Application	839 714 KB





Reštartujeme PC.

Spustíme PyCharm



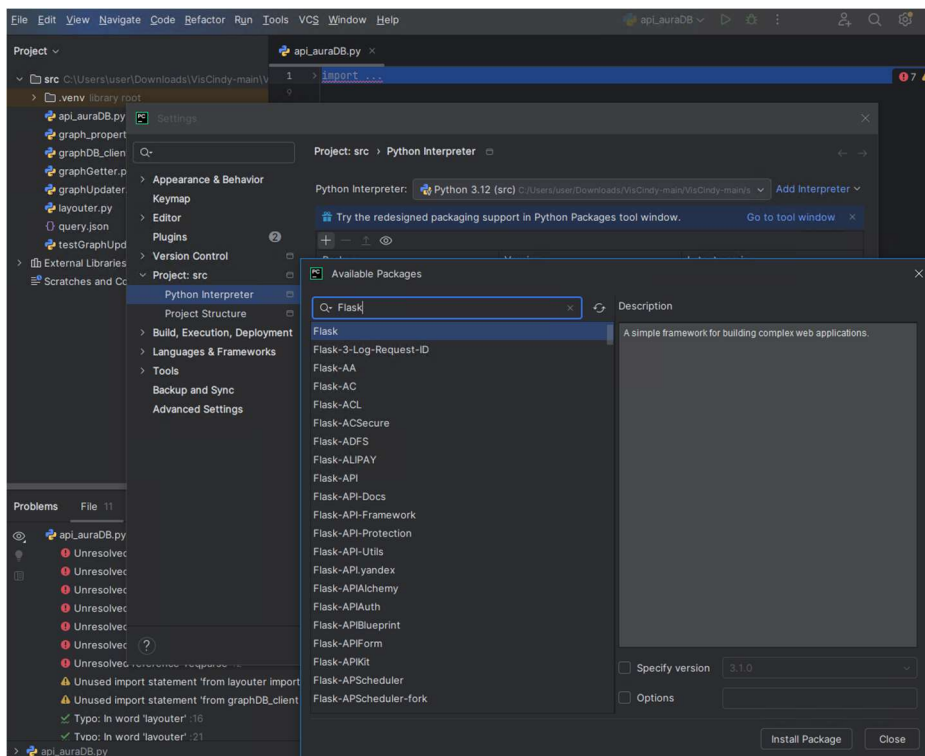
Ak by bol problém s vytvorením nového python interpetera stačí nainštalovať z tadiaľto:

<https://www.python.org/ftp/python/3.12.8/python-3.12.8-amd64.exe>

Ďalej treba stiahnuť projekt z repozitára a rozbaľiť ho u seba v počítači:

<https://github.com/MatejBudos/VisCindy.git>

HamburgerMenu -> Projects -> Settings – Python Interpreter -> Add Module



Ďalej pridáme

Flask

Flask-RestFul

Neo4J

Igraph

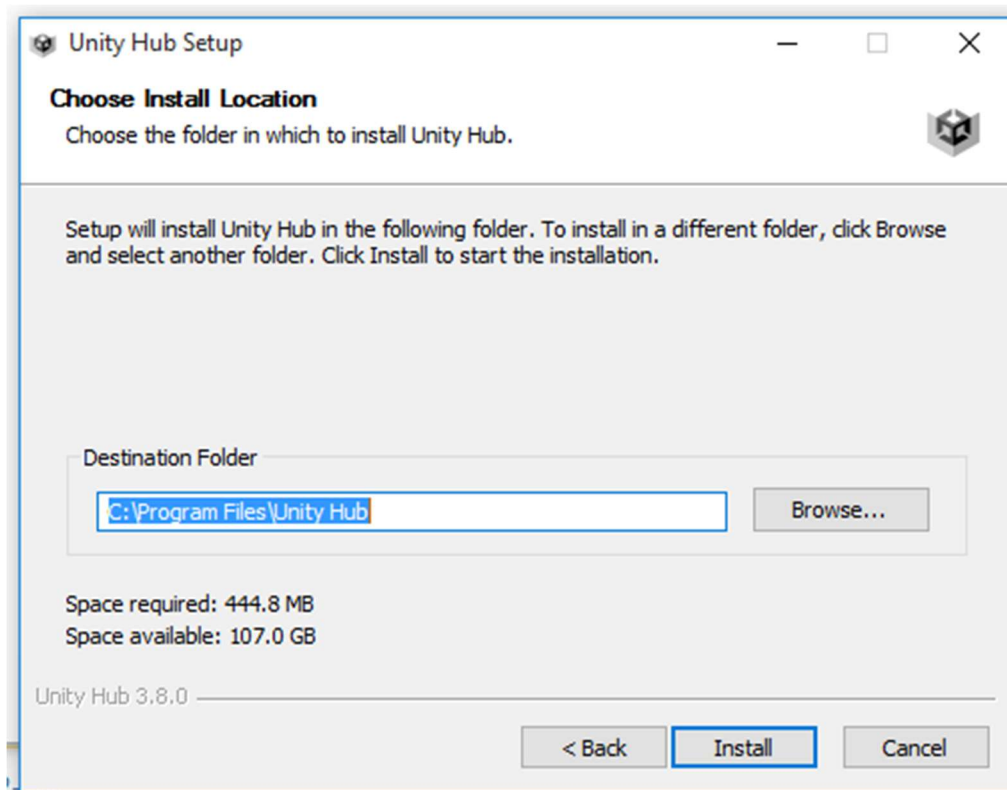
Matplotlib

Kvôli bezpečnosti sme dodatočne prekopírovali do SRC adresára authentication.json.

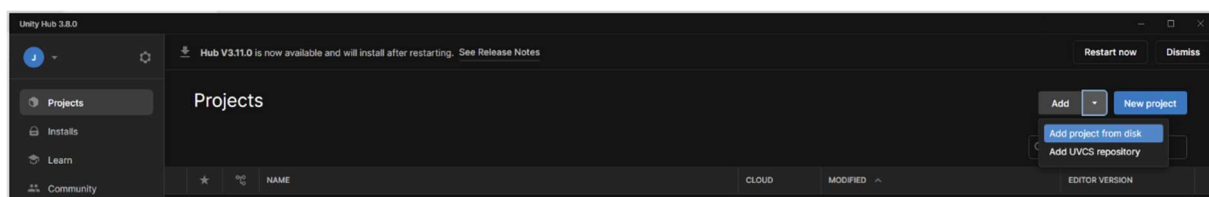
This PC > Downloads > VisCindy-main > VisCindy-main > src >				
	Name	Date modified	Type	Size
	.idea	25.01.2025 22:32	File folder	
	.venv	25.01.2025 22:34	File folder	
	api_auraDB	25.01.2025 21:45	Python File	1 KB
	authentication.json	28.11.2024 22:38	JSON File	1 KB

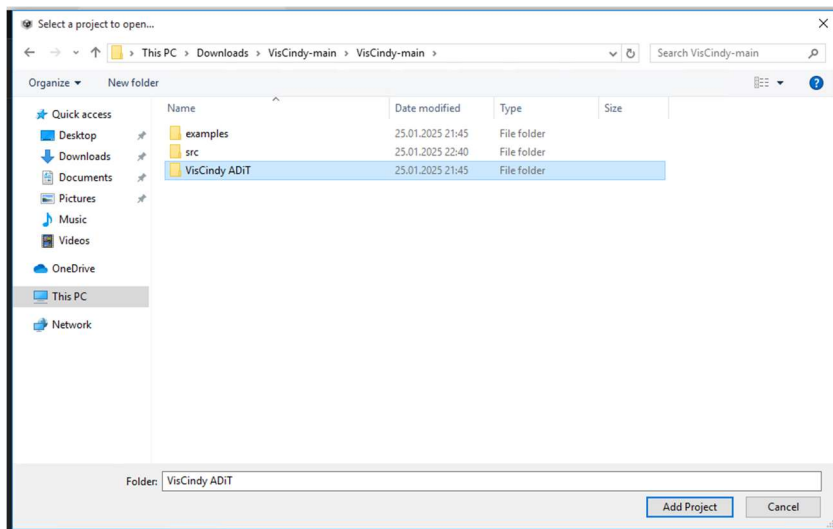
Ako posledne treba nainštalovať unity hub a k nemu unity verziu 2021.3.21f1 prípadne aj visual studio:

<https://unity.com/download>

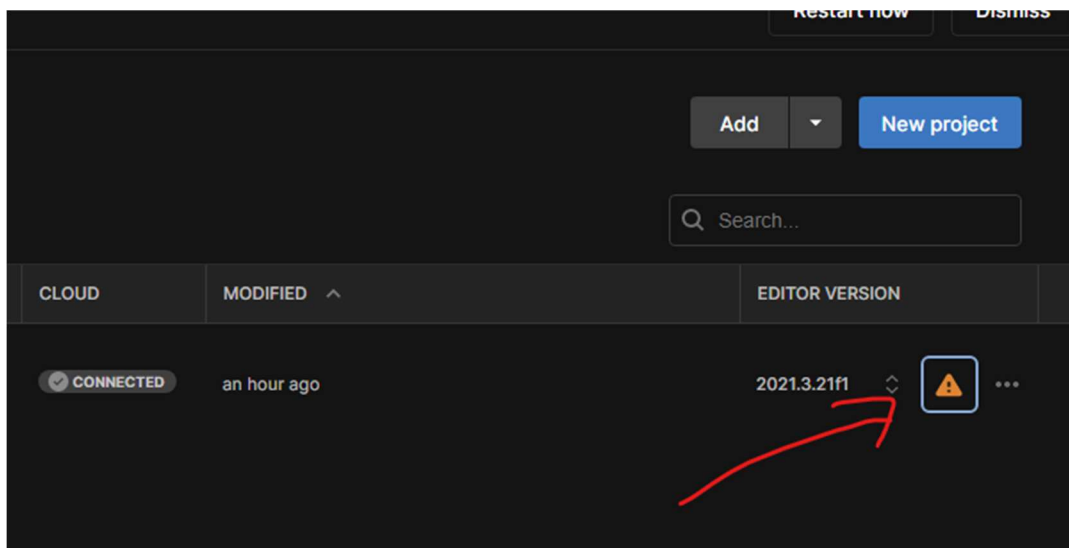


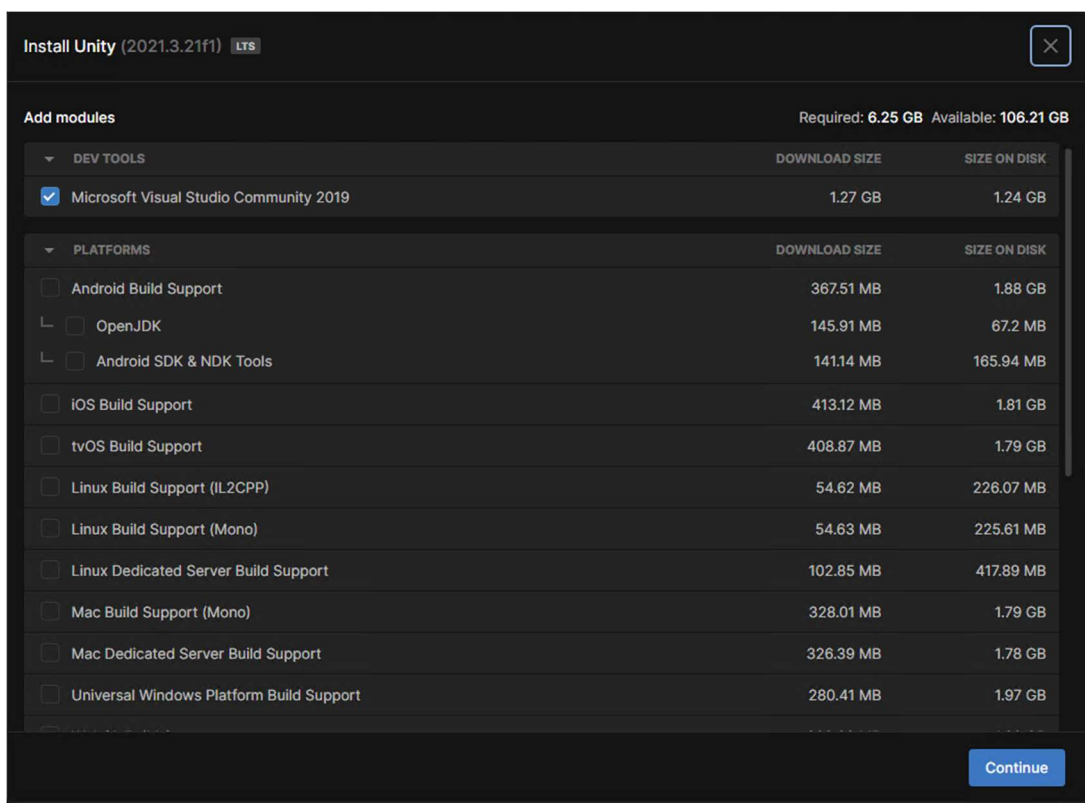
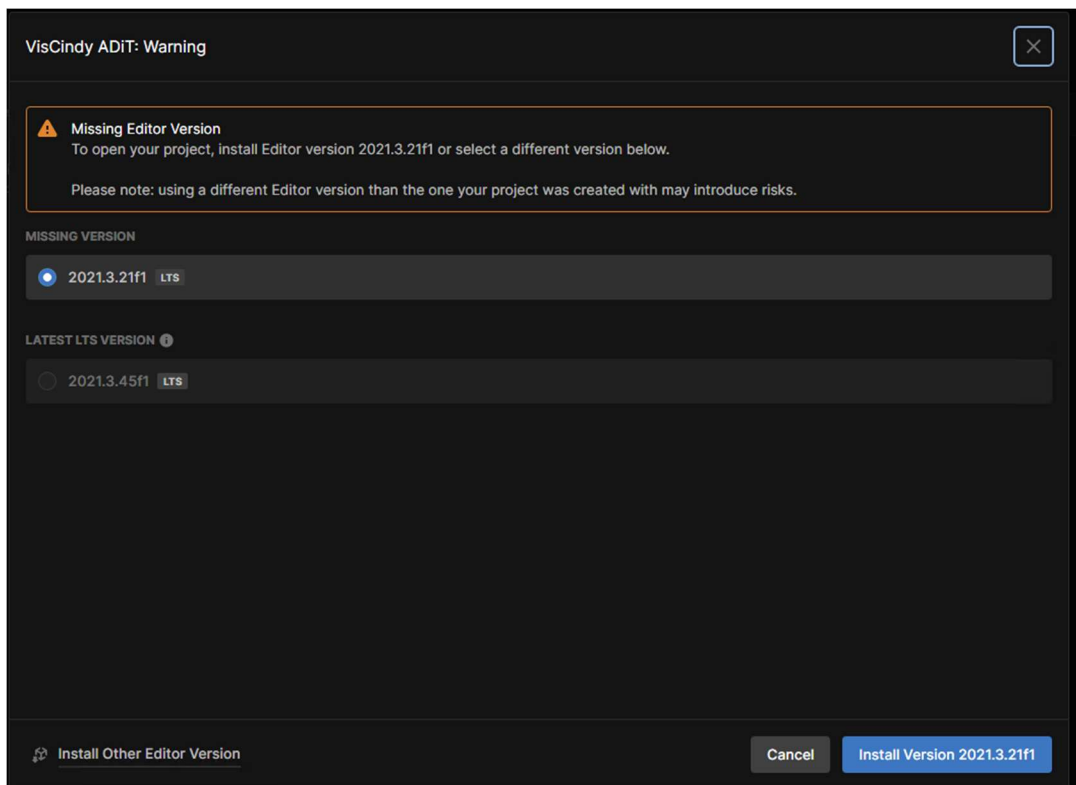
Projects -Add Projects from Disk





Klikneme na Trojuholník s vykričníkom.



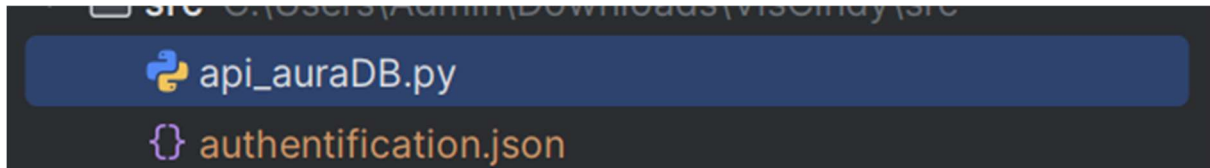


Prípadne link pre manuálne nainštalovanie Microsoft Visual Studio 2019

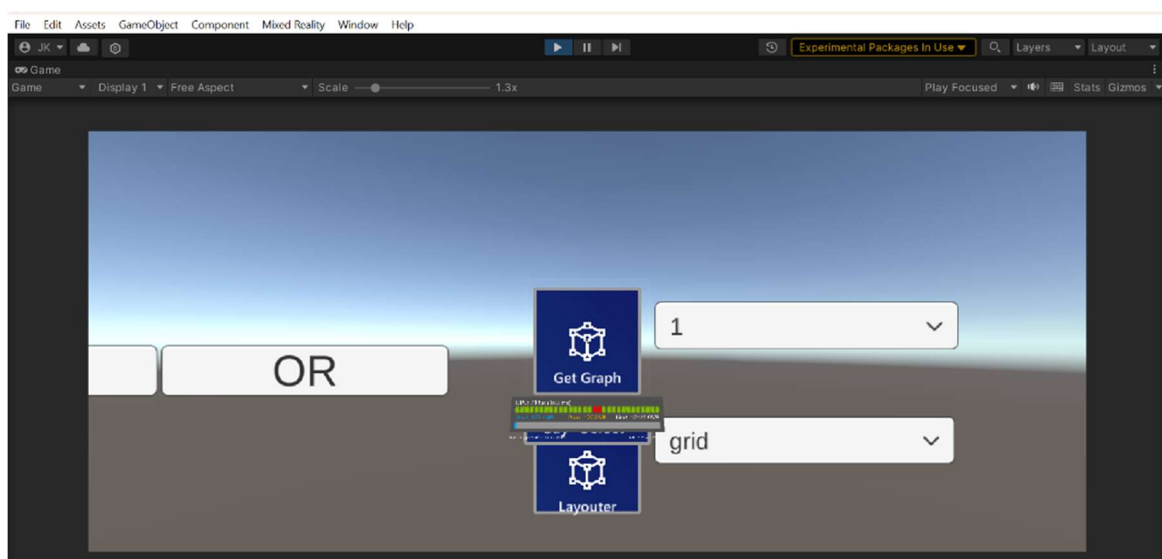
<https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSLandingPage&passive=false&cid=2030>

Používateľská príručka

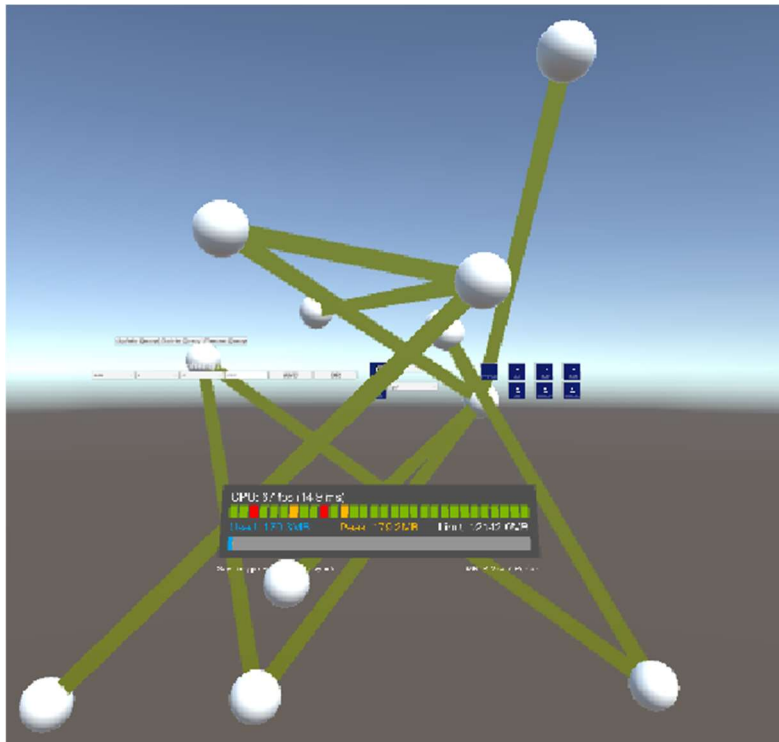
V prvom kroku si potrebujeme spustiť Aura DB v Pycharme pre načítanie grafu. Skompilujeme program `api_auraDB.py`. Pre úspešne spustenie potrebujeme mať v tom istom adresári uložený súbor `authentification.json`, ktorý v sebe obsahuje údaje na autentifikáciu, bez neho sa k Aura DB inštancii nedostaneme!



Keď už nám úspešne beží Spojenie s DB otvoríme si náš Unity Projekt a spustíme ho pomocou PlayButton tlačidla. Zobrazí sa nám takáto scéna s rôznymi ovládacími prvkami. Pohyb v scéne vieme potom vykonávať pomocou šípiek a tlačidiel PgUp a PgDn.



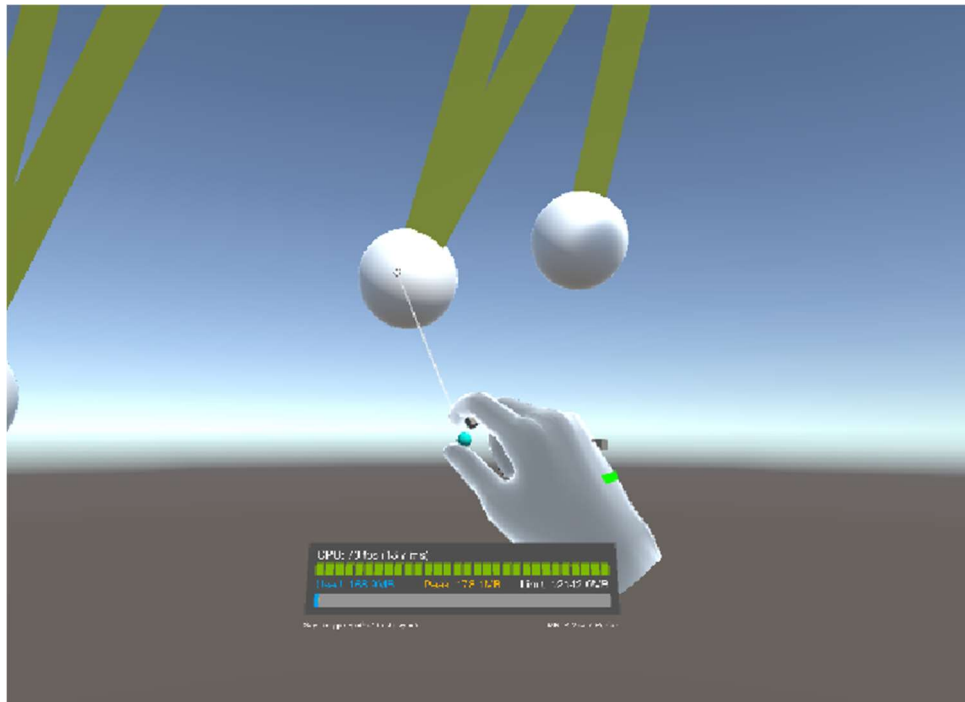
Začnime so základnými ovládacími prvkami. Vďaka vyššie položenému comboboxu si užívateľ môže jeden z grafov ktorý chce zobrazíť. V ďalšom comboboxe si môže užívateľ navoliť tvar grafu ako bude vyzeráť. Kliknutím na tlačidlo GetGraph sa takto navolený graf zobrazí.



Na hocikaké tlačidlo môžeme kliknúť iba vtedy ak sa nám pod ním zobrazí label s názvom "Say Select".



Keď už máme teda graf načítaný presunieme sa k možnostiam jeho manipulácie. Vďaka kombinácii tlačidiel Space a kliknutia ľavého tlačidla na myši vieme zobrať ľubovoľný vrchol grafu a premiestniť ho v priestore



Ďalšími z možností na manipuláciu grafu je táto paleta tlačidiel, ktorá obsahuje následné funkcionality:

1. Add Node – pridá nový vrchol do grafu na pozíciu (0,0,0)
2. Delete Node – vymaže konkrétny vrchol v grafe po zakliknutí.
3. Add Edge – po kliknutí na konkrétne dva vrcholy vytvorí hranu
4. Delete Edge – po kliknutí na dva vrcholy vymaže medzi nimi hranu ak existuje
5. UnDo – zmaže poslednú úpravu grafu
6. ReDo – vráti poslednú zmazanú úpravu funkciou UnDo
7. Commit changes – všetky zmeny v grafe sa pošlú na vykonanie do DB

Update Query Delete Query Finalise Query Post Query

WHERE ((x > 23 OR description = nice node) AND (y < 44 OR label != not))

☒ Node/Vertex

Node	x	>	23	AND	OR
Node	y	<	44	AND	OR
Node	description	=	nice node	AND	OR
Node	label	!=	not	AND	OR

Ako posledná funkcionálna je vizuálny Cypher Query Builder, ktorý umožňuje používateľovi vytvárať Cypher dotazy pomocou vizuálneho rozhrania bez potreby manuálneho písania syntaxe.

Hlavné funkcionality:

1. Pridávanie podmienok:

- V riadku s comboboxmi môže používateľ vybrať:
 - Typ uzla (napr. Node/Vertex),
 - Atribút (napr. x, y, description, label) vytiahnutý z databázy pre konkrétny graf,
 - Operátor (napr. >, <, =, !=),
 - Hodnotu podmienky (napr. číslo alebo text).
- Po nastavení jednej podmienky môže používateľ použiť logickú spojku (AND alebo OR), čím sa pridá nový riadok na vytvorenie ďalšej podmienky.

2. Tlačidlo "Update Query":

- Po kliknutí zobrazí aktuálny stav podmienok v textboxe vo formáte ako napríklad:
((p0 OR p1) AND p2), kde p0, p1, p2 označujú jednotlivé podmienky.

3. Tlačidlo "Delete Query":

- Vymaže všetky zadané podmienky a resetuje query builder do pôvodného stavu.

4. Tlačidlo "Finalise Query":

- Vytvorí finálnu verziu dotazu vo formáte Cypher, kde sa symboly p0, p1 nahradia skutočnými podmienkami. Napríklad:
WHERE ((n.x > 23 OR n.description = 'nice node') AND (n.y < 44 OR n.label != 'not')).

5. Tlačidlo "Post Query":

- Odošle vytvorený Cypher dotaz priamo na databázu na spracovanie.