2024

IS Bedmintonového centra

PROJEKT VIS

BELOS ADAM BEL0130

1. Vízia

Prečo?

Tento IS Umožní zákaznikom rezervovať sa online namiesto rezerácie po telefóne. Odľahčí to prácu zamestnancov a objednávky nebudú zapísané v knížke a budú verejne dostupné na internete.

Čo?

Informačný systém bedmintonového centra. Bude to online rezervačný systém kde sa bude dať zarezervovať si kurt na jednu hodinu alebo kontrolovať historiu rezervácií.

Kto?

Informačný systém je primárne určený pre zákaznikov ktorý sa môžu objednať na jednotlivé kurty.

Kde?

-Webová aplikácia na vlastnej domene. Prístup všade na internete.

Kedy?

Keď si chce zákazník zahrať bedminton a chce si rezervovať kurt v hale. Takže prístupnosť systému bude non stop.

Ako?

Webová aplikácia na vlastnej domene ktorá bude spravená formou html formulárov a servérová časť v c# s postgresql.

2. Funkčná špecifikácia

Create Account: vytvorenie účtu uživateľa

Login: prihlásenie sa do účtu, zároveň ak je uživateľ už prihlásený na účte odhlási ho z prihláseneho zariadenia, a následne prihláseny na novom zariadení.

Show Reservations per date: zobrazí rezervácie podľa zvoleného dátumu.

Create Reservation: vytvorenie rezervácie podľa prihláseného účtu

Delete Reservation: vymazanie rezervácie podľa prihláseného účtu

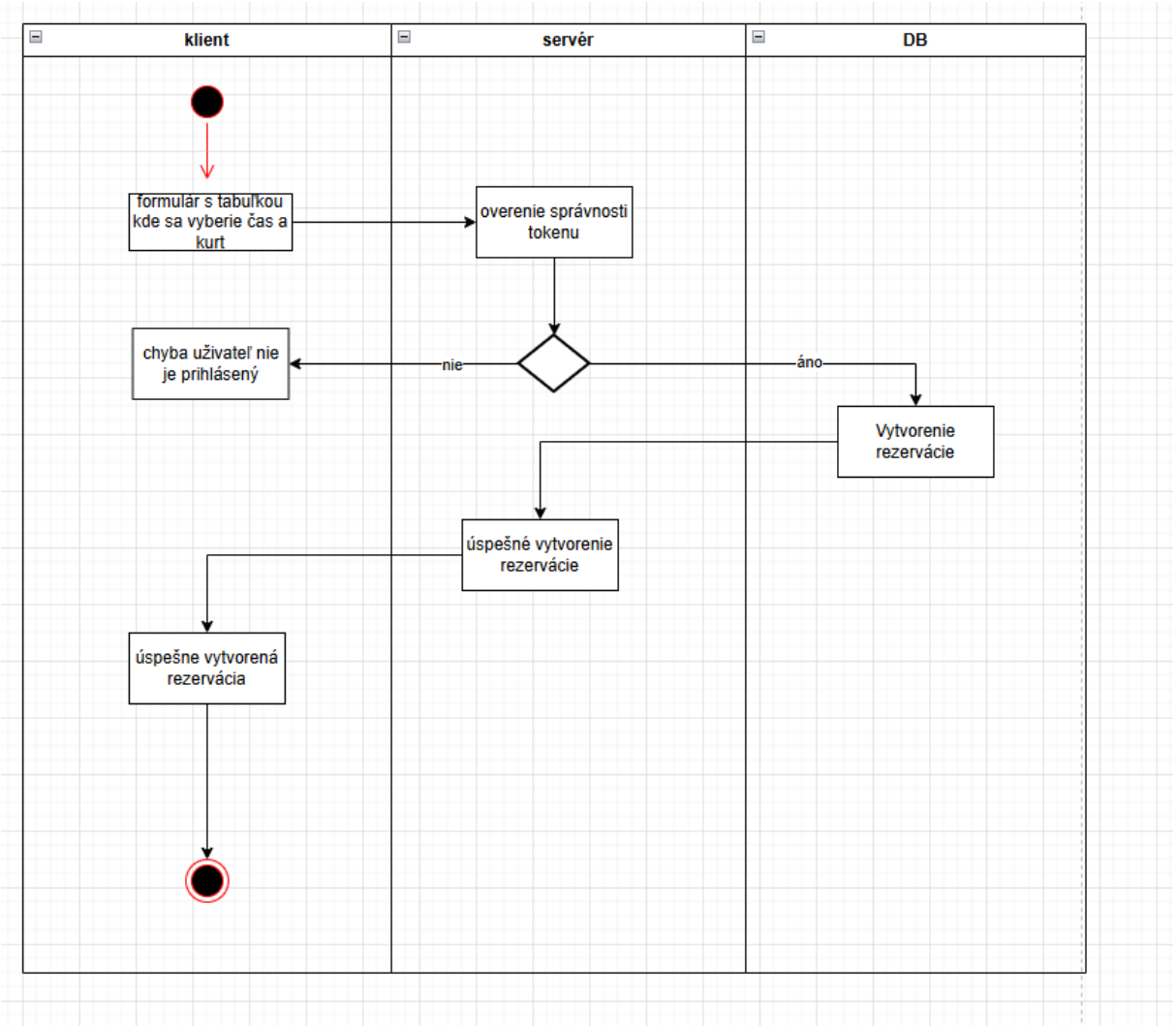
Show Own Reservations: zobrazenie rezervácií podľa prihláseného účtu.

User: prihlásený uživateľ

Client: neprihlásený uživateľ

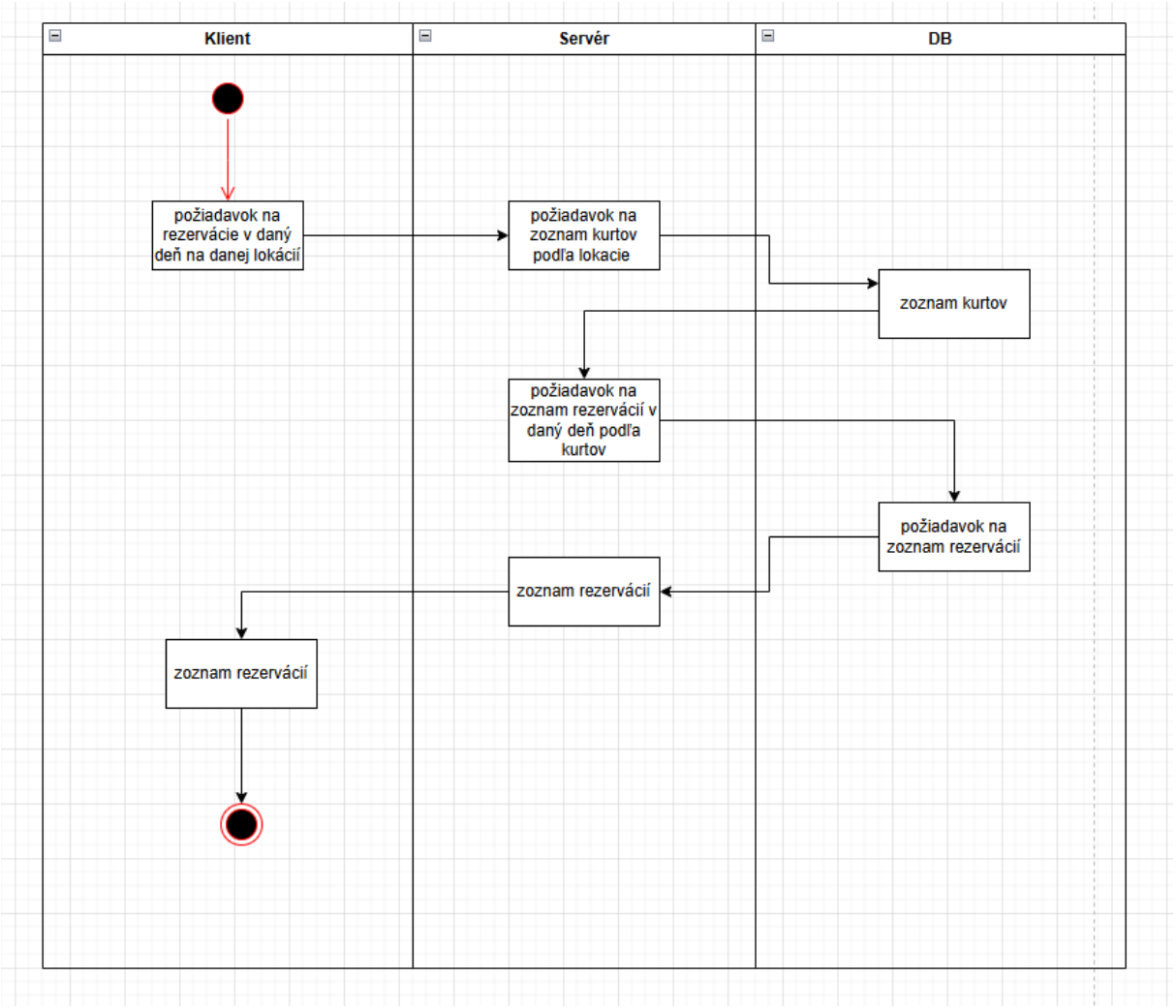
Usecase vytvorenie rezervácie activity diagram

Overuje či je uživateľ naozaj prihlásený a následne pre daný účet vytvára rezerváciu. Využíva sa keď si uživateľ chce vytvoriť rezerváciu.



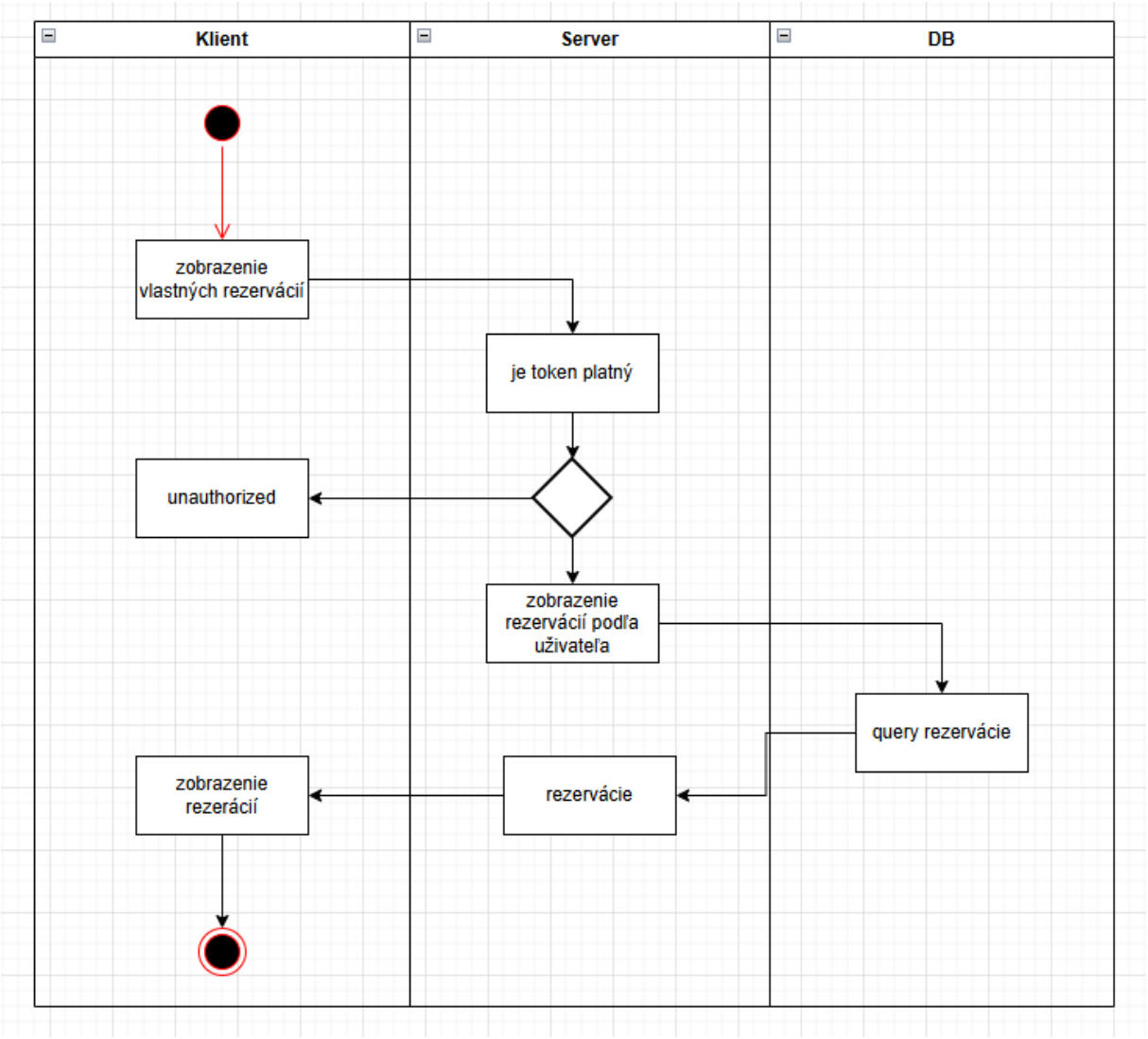
Usecase zobrazenie rezervácií activity diagram

Zobrazenie rezervácií podľa dátumu a lokácie. Zobrazí sa pri zobrazení formuláru vytvorenia rezerácie ako súčasť tabuľky s rezerváciami



Usecase zobrazenie vlastných rezervácií activity diagram

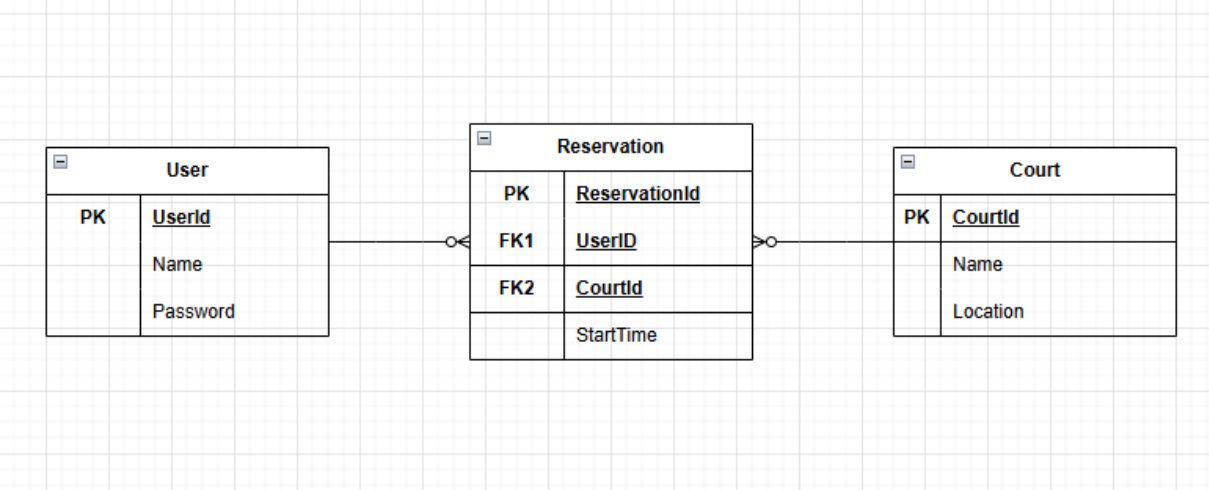
Zobrazenie rezervácií podľa uživateľa. Využije sa keď je uživateľ prihlásený a chce si zobraziť vlastné rezervácie



3. Technická specifikace

Doménový model

Doménový model pozostáva z 3 tabuliek aj keď som zvažoval použiť 4. Tabuľku na ktorá by ukladala tokeny prihlásených uživateľov ale nakoniec som sa rozhodol vhľadom na množstvo používateľov rozhodol tokeny Cachovať.



Predpokladaný počet a Velikosti entit

Celková veľkosť dat je veľmi malá 505,3KB.

Entita	Predpokladaný počet	Veľkosť jednej	Veľkosť všetkých
User	400	260B(128B+128B+4B)	104KB
Reservation	20000	20B(3*4B+8B)	400KB
Court	5	260B(128B+128B+4B)	1,3KB

Použité technológie

Databáza

Zvážoval som MSSQL a Postgresql. Vzhľadom že som chcel prezentovať funkčný systém s datami na inom počítači ako som chcel vyvíjať tak hľadal možnsti mimo lokálnej databázy. MsSQL som používal s azure cloudu ale už mám vypotrebovaný balíček ktorý mi umožnuje zdarma vytvoriť databázu. Zamýšľal som sa nad školskou databázou ale zakaždým byť pripojený ku školskej VPN a riešiť prípadné výpadky sa mi nechcelo. Následne som premýšľal nad použitim Railways Postgresql databázu ale vzhľadom že na to že tam už mám databázový server ktorý využívam a nechcel som si priplácať za ďalší server som sa rozhodol pre inú službu. Nakoniec som sa rozhodol použiť aiven s Postgresql na AWS free verziu.

Servér

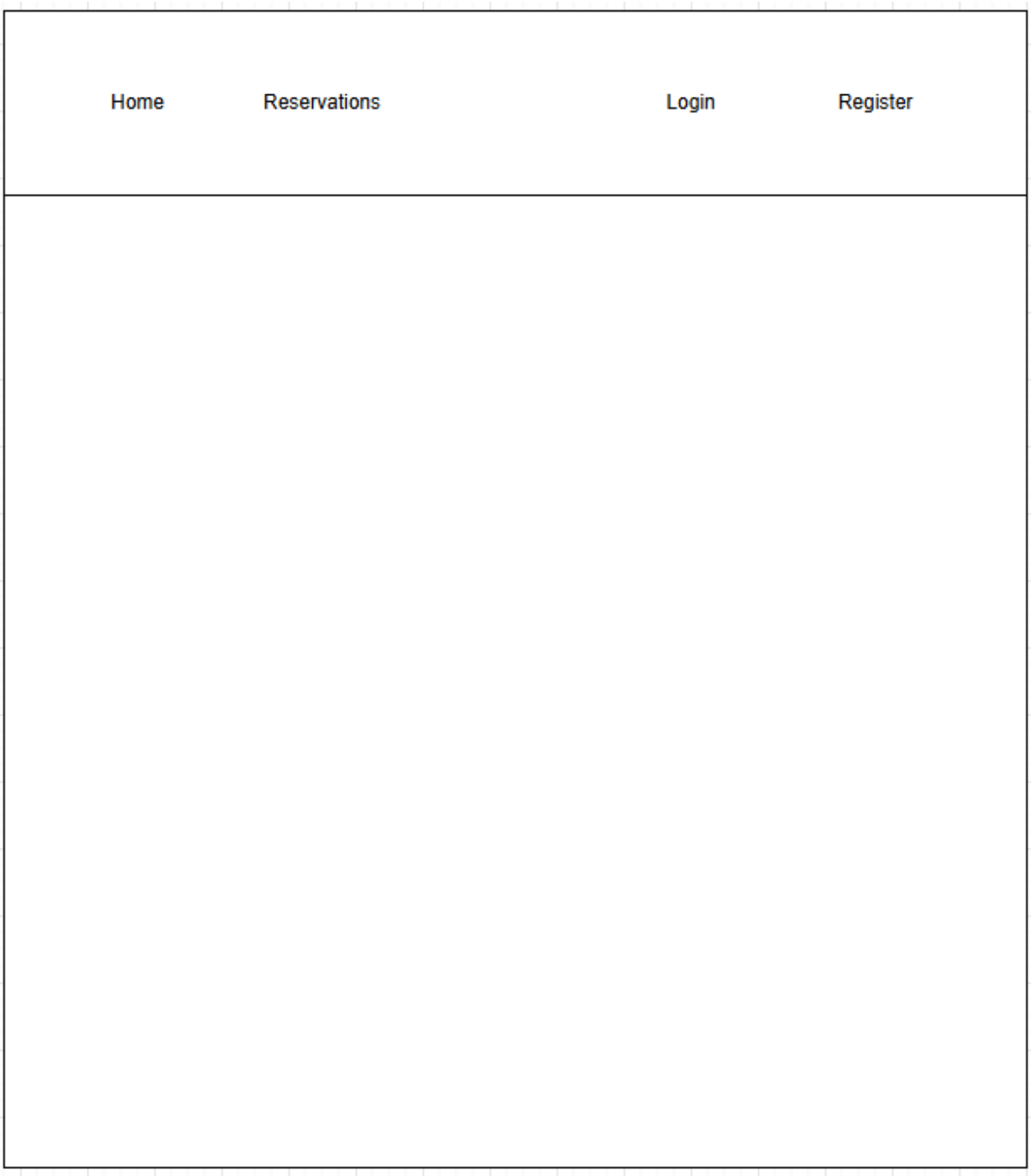
Pre mňa celkom no brainer C# dotnet. Ďalšie možnosti python alebo java alebo c prípadne Next.js alebo obdobné. Zvykol som si používať c#, kod je prehľadný a hlavne sa v ňom ľahko hľadajú chyby.

Klientská strana

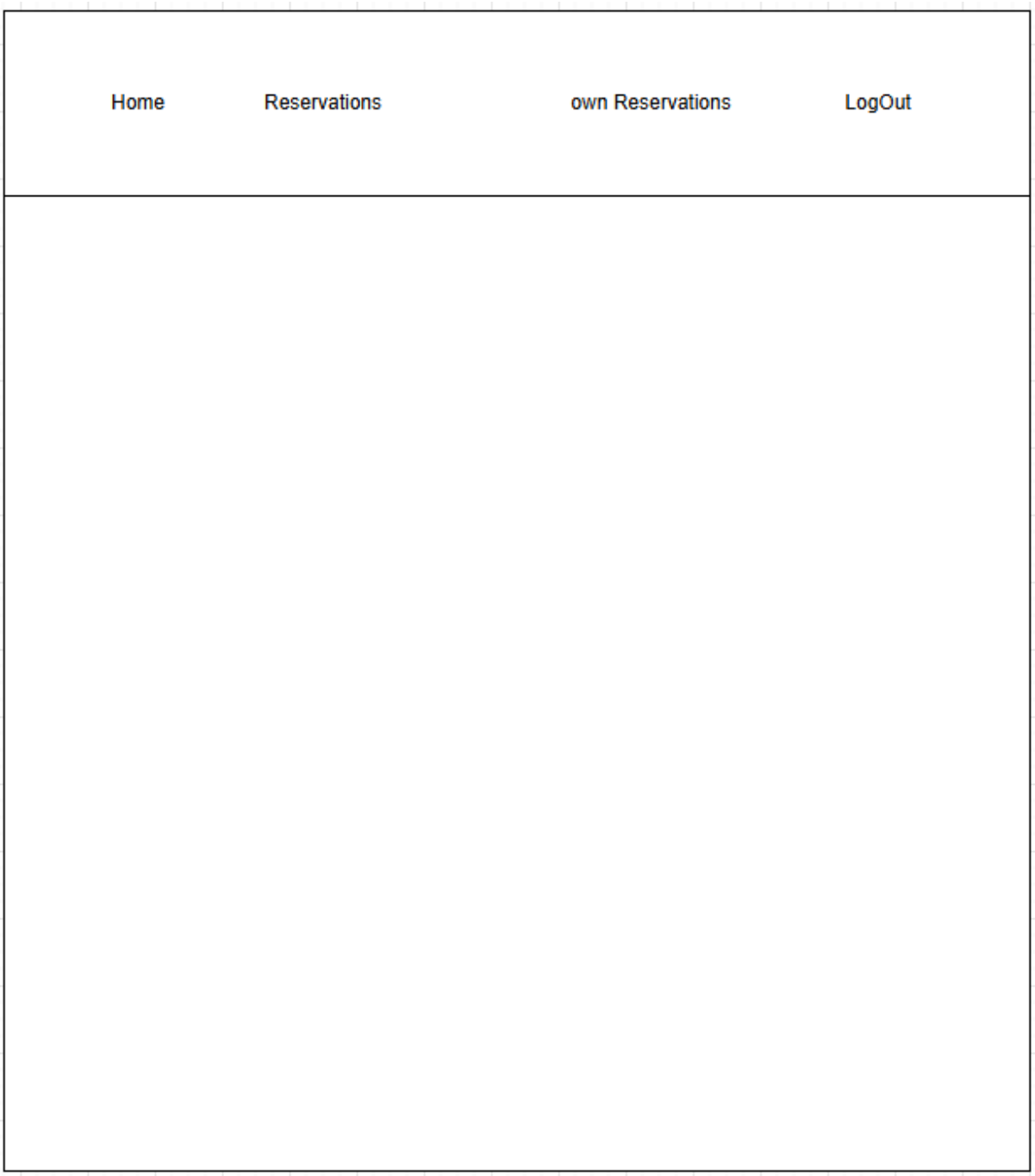
Rozhodovanie či sa použije niaka knižnica na frontend alebo sa bude používať čistý js. Zvažoval som použiť react alebo angulár alebo vue.js alebo jquery. Najviac ma lákal react vzhľadom na to že je to už doba čo som v ňom nevyvíjal ale nakoniec som sa rozhodol použiť samostaný js aj napriek tomu že som chcel aby klientska strana mala niektoré výhody SPA. Všetky stránky budú serverside render ale prihlasovanie a prechádzanie rezervácií sa musí javiť ako klientside render kde sa data budú brať s API. Vzhľadom na relatívnu jednoduchosť projektu je čistý js jednoduché riešenie ktoré je aj veľmi efektívne. Pre posielanie requestov na API som zamýšľal použiť axios ale nakoniec barebones js fetche zvíťazili. Serverside render je pomocou MVC.

4. wireframe

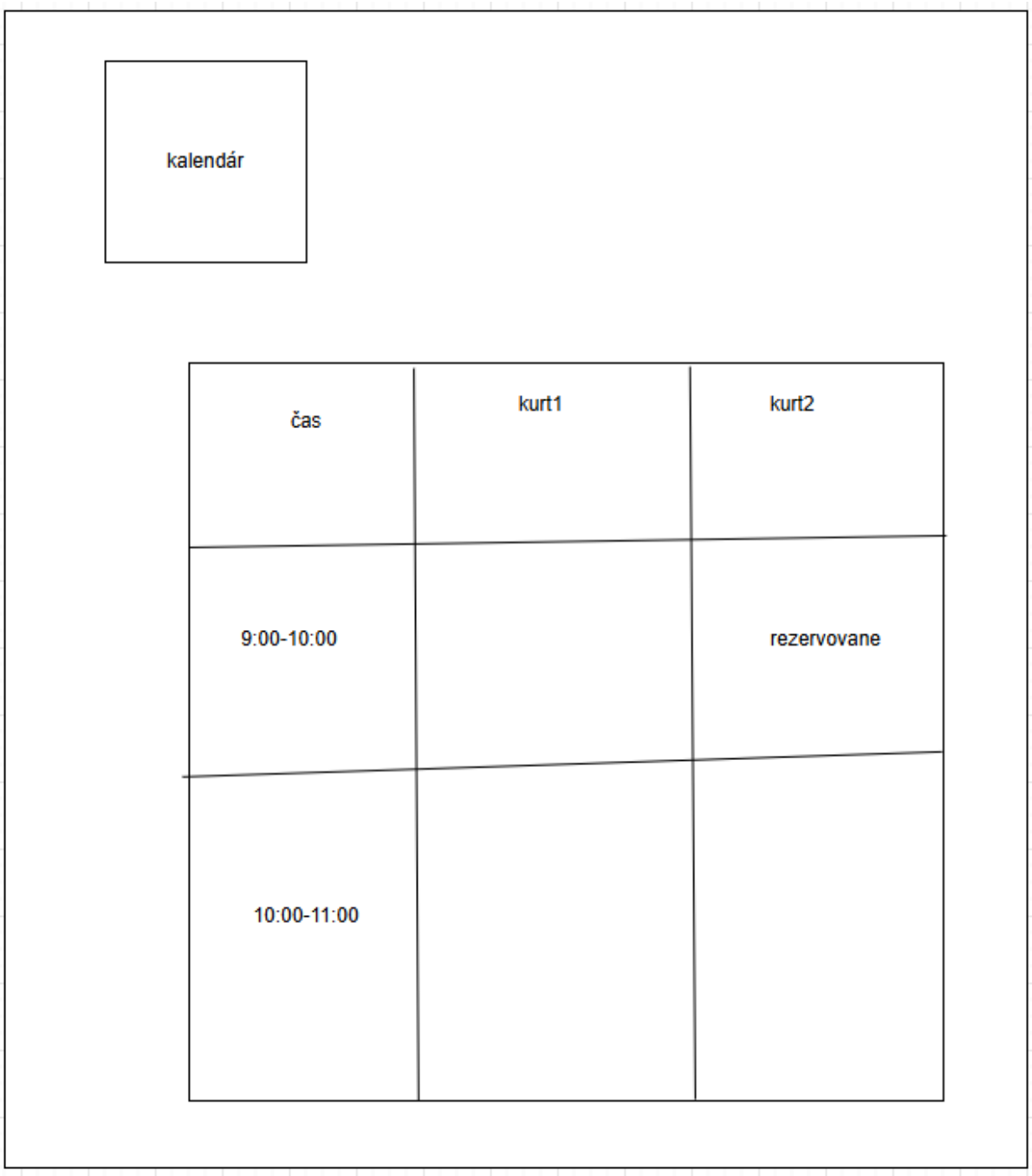
Layout neprihlásený



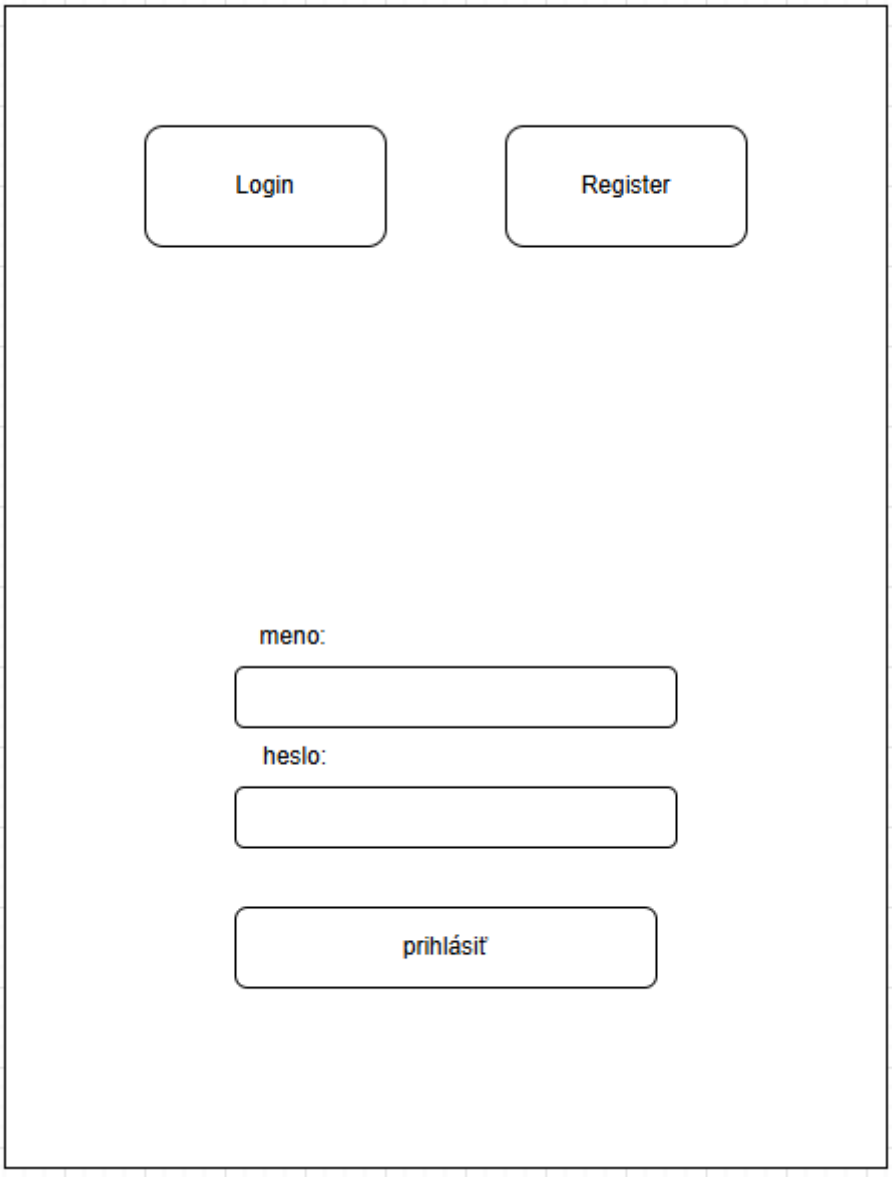
Layout prihlásený



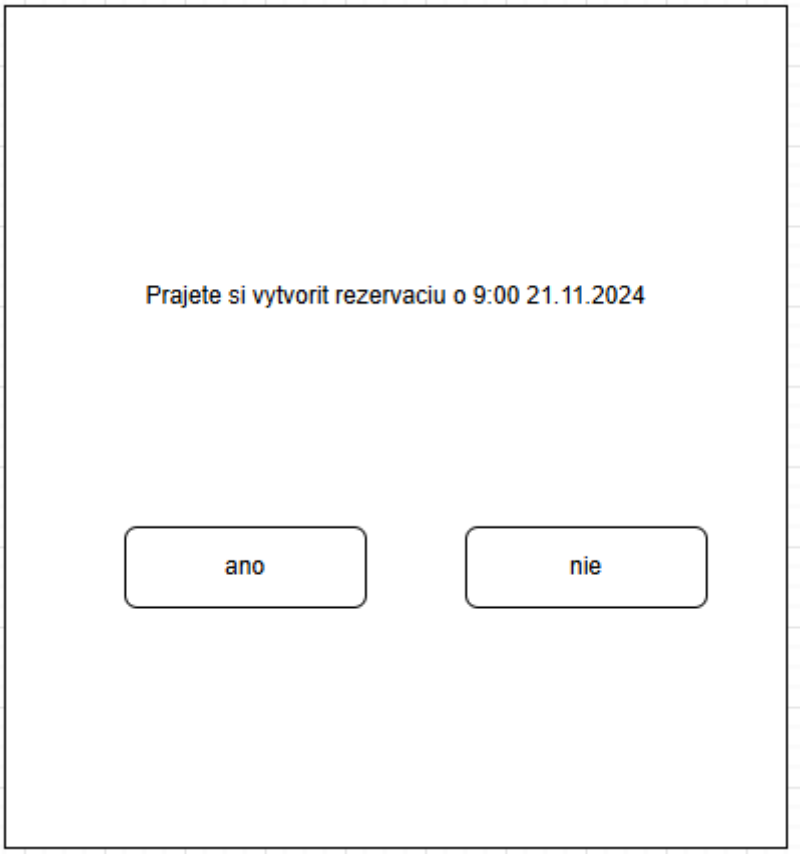
Tabuľka rezervácií



Login/Register Modal nad backdropom



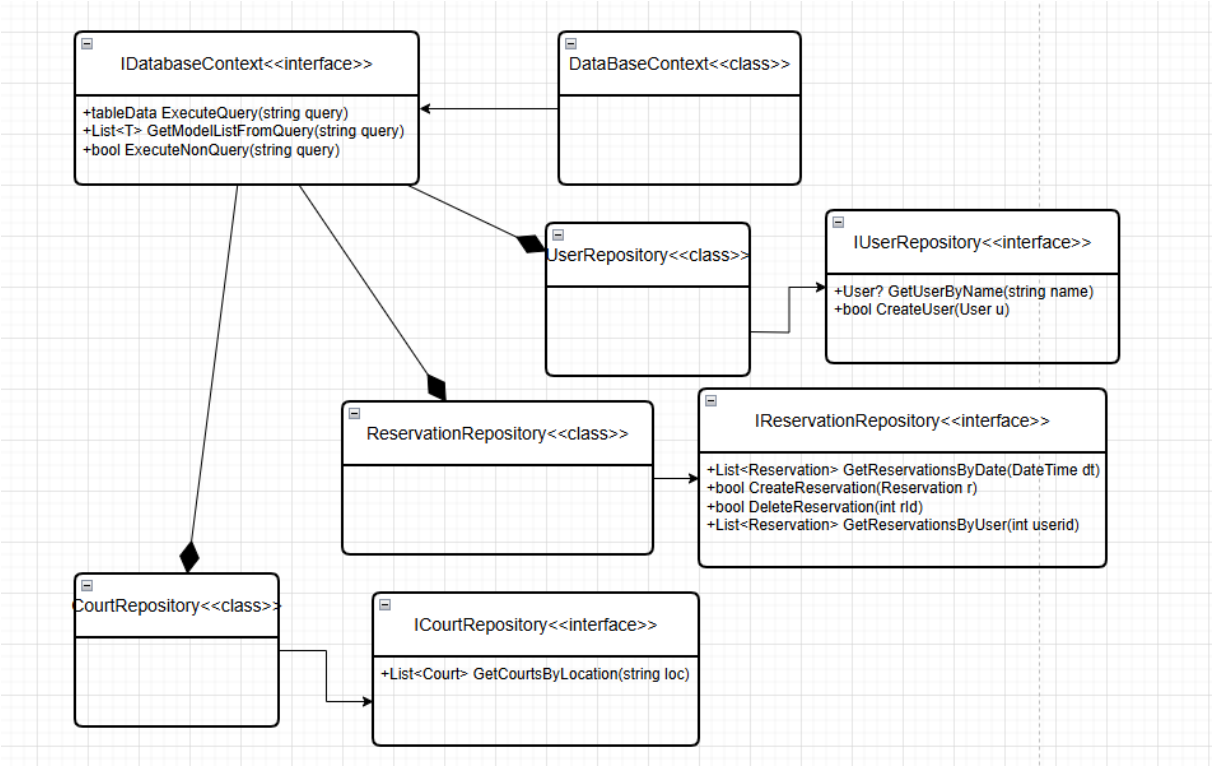
Create/delete reservation modal nad backdropom



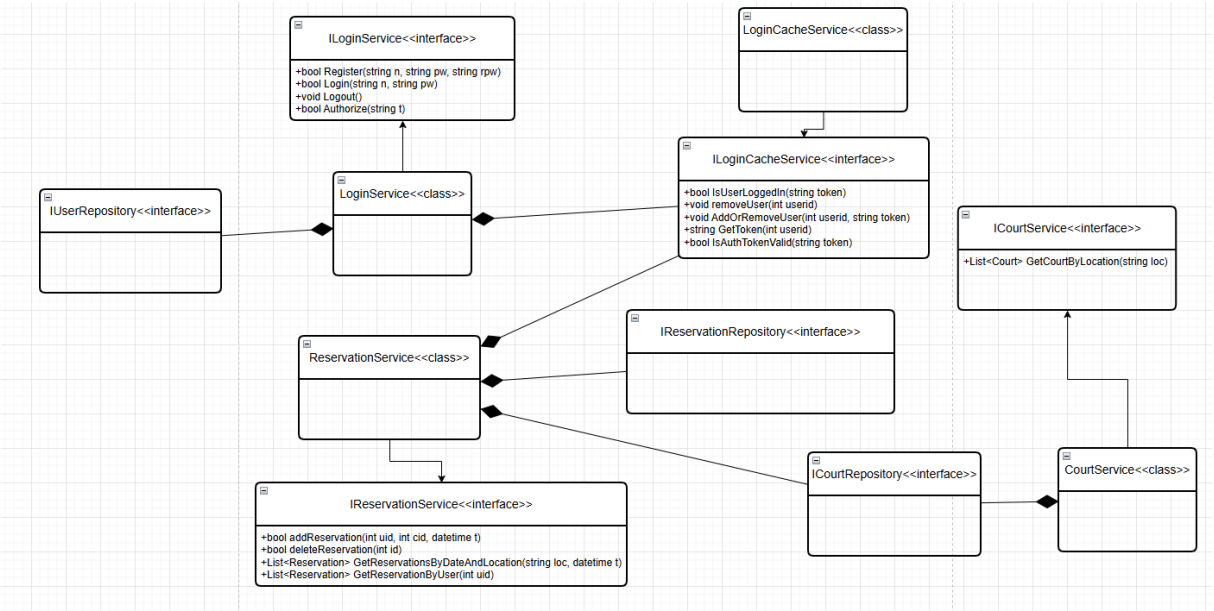
5. návrh doménového modelu

Triedny diagram

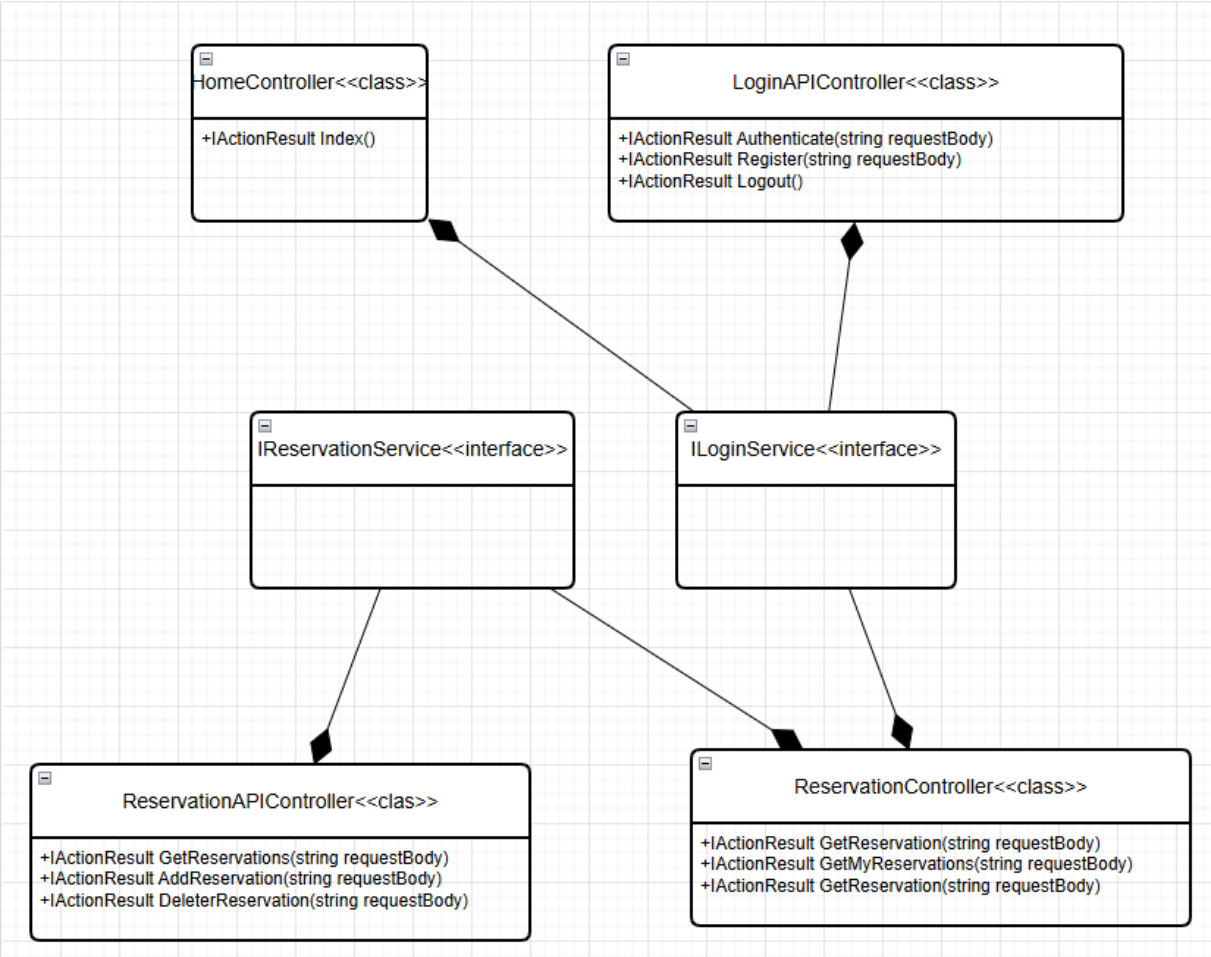
DataAccessLayer



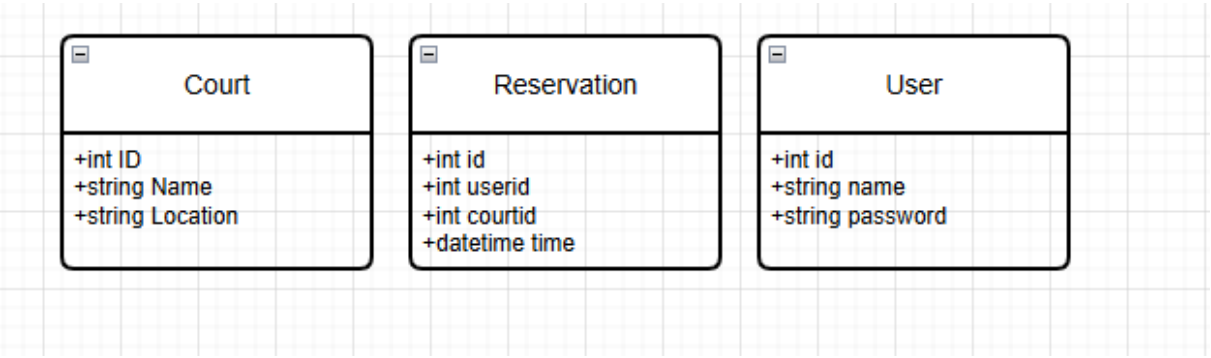
ApplicationLayer



PresentationLayer



Domenove modely



Použité návrhové vzory

Dependency Injection

Pre napojenie komponent od databázového kontextu ku jednotlivým kontrollerom.

Singleton

Cacheovanie prihlásených uživateľov, je nutné aby mali rovnaké data všetky requesty.
Databázový kontext je tiež singleton

Mapper

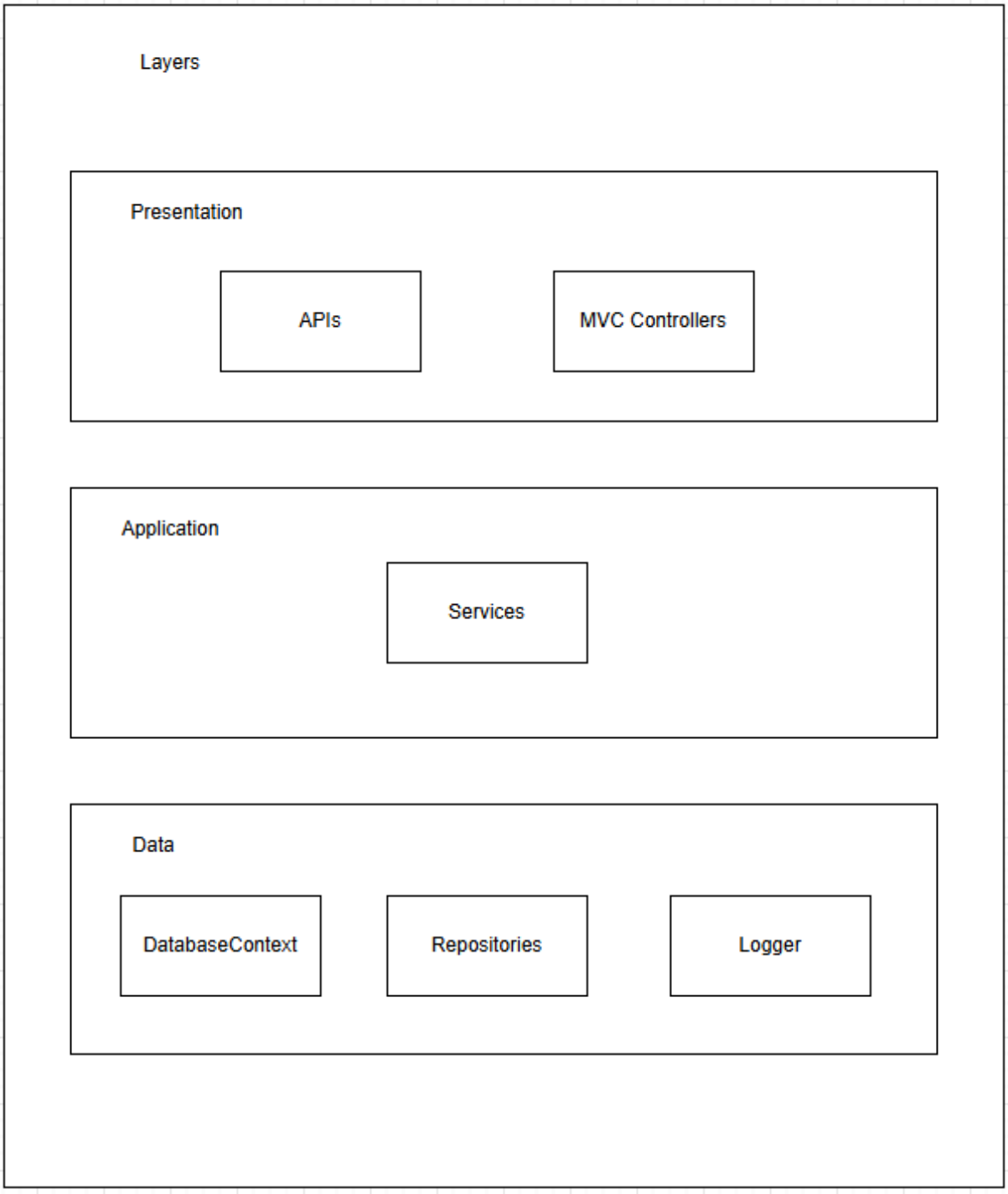
Generické mapovanie z tabuľky na doménové modeli

Iterator

Prechádzanie jednotlivých tabuľkových atributov v generickom mapovaní tabuľky.

6. Popis architektury systemu

Diagram Component



Topologia nasadenia