Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Hlídač cen kolekce Magic: The Gathering karet

2023

Tomáš Nádvorník

Vedoucí práce:
Mgr. Radek Janoštík, Ph.D.

Studijní obor: Informatika, prezenční forma

Bibliografické údaje

Autor:	Tomáš Nádvorník
Název práce:	Hlídač cen kolekce Magic: The Gathering karet
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2023
Studijní obor:	Informatika, prezenční forma
Vedoucí práce:	Mgr. Radek Janoštík, Ph.D.
Počet stran:	27
Přílohy:	1 CD/DVD
Jazyk práce:	český

Bibliograhic info

Author:	Tomáš Nádvorník
Title:	Magic: The Gathering card collection price monitor
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2023
Study field:	Computer Science, full-time form
Supervisor:	Mgr. Radek Janoštík, Ph.D.
Page count:	27
Supplements:	1 CD/DVD
Thesis language:	Czech

Anotace

Magic: The Gathering je sběratelská karetní hra rozšířená po celém světě. Pokud chcete hrát na profesionální úrovni, musíte neustále rozšiřovat svou sbírku o nové karty. Cena karet není regulována, proto může ze dne na den dojít k obrovským výkyvům. Cílem práce je ušetřit sběratelům peníze při nákupu nových karet a navýšit zisky při jejich prodeji, a to tvorbou webové aplikace. Pro získání informací o jednotlivých kartách je využito veřejné API scryfall.com. Aplikace periodicky kontroluje ceny karet a vykresluje historii do grafu. Uživatel aplikace si může nastavit cenový výkyv, na který bude upozorněn emailem.

Synopsis

Magic: The Gathering is a collectible card game spread all over the world. If you want to play at a professional level, you need to constantly expand your collection with new cards. The price of the cards is not regulated, so there can be huge fluctuations from day to day. The goal of this thesis is to save collector's money when buying new cards and to raise the profit when selling by creating a web application. Public API scryfall.com is used to get information about the cards. The application will periodicaly check the prices of the cards and plot their history in to a chart. User of the application can adjust the price change of which he'll be informed via email.

Klíčová slova: Magic: The Gathering, SvelteKit, ASP .NET, Microsoft SQL Server

Keywords: Magic: The Gathering, SvelteKit, ASP .NET, Microsoft SQL Server

Děkuji vedoucímu mé práce Mgr. Radku Janoštíkovi, Ph.D. za ochotu a vedení a své rodině za korekturu a podporu.

Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

datum odevzdání práce podpis autora

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Magic: The Gathering je sběratelská karetní hra, která byla vytvořena matematikem Richardem Garfieldem a vydána společností Wizards of the Coast v roce 1993. Hra zahrnuje dva nebo více hráčů, kteří používají balíčky karet představujících magická kouzla, bytosti a artefakty k poražení svých protivníků.

V roce 1996 zahájili Wizards of the Coast Pro Tour, sérii turnajů pořádaných po celém světě, která každý rok vyvrcholí mistrovstvím světa. Kompetetivní hraní vyžaduje vysoce precizní výběr karet. Jednotlivé karty se často opakují, aby se maximalizovala konzistentnost. To hráče často vede k nákupu samostatných karet namísto tradičního rozšiřujícího balíčku, který obsahuje náhodné karty.

Cena karet se odvíjí od jejich vzácnosti, hratelnosti a stavu. Například karta Black Lotus z edice Alpha v nehraném stavu byla v roce 2021 vydražena za 511 tisíc dolarů. Kromě toho má na cenu karty vliv i aktuální žádanost. Pokud je o kartu velký zájem, například díky nově objevené strategii, může se její cena ze dne na den zestonásobit.

Ať už sběratel nebo profesionální hráč, oběma by se hodilo kdyby byli upozorněni na cenové výkyvy u karet, o které se zajímají. Tuto potřebu by šlo naplnit webovou aplikací, ve které by uživatel vybral na jaký cenový výkyv chce být u jednotlivých karet upozorněn.

2 Technická dokumentace

Aktuální informace o kartách lze získat z veřejného API scryfall.com. Neposkytuje ovšem žádné informace o minulosti. Scryfall.com má restrikce na počet požadavků za sekundu. Aby se predešlo zákazu, jsou všechny informace potřebné k funkci aplikace uloženy v databázi. Aby uživatelé mohli vidět vývoj ceny, je potřeba každý den kontrolovat ceny všech karet a změny uložit.

2.1 Použité technologie

Technologie byly vybrány po špatné zkušenosti v první verzi projektu.

Klientská část aplikace byla původně implementována v Blazoru, který umožňuje vývojářům vytvářet webové aplikace pomocí C#a .NET namísto tradičních jazyků pro vývoj webu, jako je JavaScript, HTML a CSS. To ovšem za cenu nepříjemného a pomalého vývoje.

Pro definování databáze a komunikaci s ní byl původně použit Entity Framework Core, což je ORM vyvinutý společností Microsoft pro aplikace .NET. Umožňuje vývojářům pracovat s relačními databázemi pomocí objektů .NET, čímž eliminuje potřebu přímo psát dotazy SQL. To ovšem za cenu rychlosti, která v případě tohoto projektu, byla velmi nízká.

2.1.1 SvelteKit

SvelteKit[1] je postavený na Svelte[2]. Javascript frameworku, který používá kompilátor místo virtuální DOM, jak je tomu například u React[3] nebo Vue[4]. Svelte vytváří reaktivní graf závislosti mezi různými komponentami aplikace. Když se stav komponenty změní, Svelte automaticky aktualizuje dotčené části uživatelského rozhraní, aniž by bylo nutné ručně aktualizovat nebo znovu vykreslovat stránku. SvelteKit navrch Svelte přidává routing pro možnost definování více stránek a Server Side Rendering (SSR) pro možnost komunikace s dalšími aplikacemi.

2.1.2 Microsoft SQL Server

Microsoft SQL Server je systém pro správu relačních databází vyvinutý společností Microsoft. Nabízí možnost uložených procedur, díky kterým lze nadefinovat rozhraní pro interakci s databází. Není tedy třeba psát vlastní SQL dotazy, ale stačí zavolat uloženou proceduru.

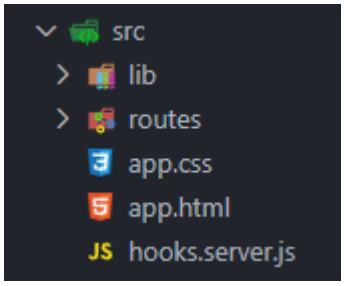
2.1.3 ASP.NET Core

ASP.NET Core je platforma pro budování webových aplikací a služeb pomocí .NET a C# vydaná společností Microsoft v roce 2016 jako nástupce dřívějšího APS.NET frameworku. Lze provozovat na různých platformách, včetně Windows, Linuxu a macOS, a integruje se s širokou škálou nástrojů a knihoven pro vývoj webu. V tomto projektu je použita verze 7.0. Pro komunikaci s databází je použito micro-ORM Dapper.

2.2 Struktura klientské části projektu

Na obrázku 1 můžeme vidět strukturu klientské části projektu.

- `routes` obsahuje jednotlivé stránky.
- `lib` obsahuje nástroje a komponenty.
- `app.css` definuje globální styly.
- `app.html` definuje šablonu stránky.
- `hooks.server.js` obsahuje funkci, která se spustí pokaždé, když server SvelteKit obdrží požadavek – ať už se to stane za běhu aplikace, nebo během předběžného vykreslování – a určí odpověď. Tato funkcionalita umožňuje determinovat, jestli je uživatel přihlášen nebo ne, aniž bychom se zabývali tím, na které jsme stránce.



Obrázek 1: Struktura klientského projektu

2.2.1 Lib

Na obrázku 2 můžeme vidět strukturu adresáře `lib`.

2.2.1.1 adresář components

Obsahuje Svelte komponenty. Ty jsou definovány jednak pomocí bloku `<script>`, který obsahuje JavaScript kód komponenty, jednak pomocí bloku <style>, který obsahuje CSS kód komponenty. HTML kód komponenty je definován v šabloně samotné, ta může zahrnovat dynamické výrazy a datové vazby. Komponenta také může mít vlastnosti, jež slouží k předávání dat mezi komponentami.

Každá stránka aplikace je definována právě takovou komponentou. Komponenty v tomto adresáři však stránky nejsou. Jsou definovány za účelem znovupoužití na více stránkách nebo pro posunutí složitosti ze stránky samotné.

Jedním z cílů práce bylo klást důraz na uživatelské rozhraní. Zásadní funkcionalitou aplikace je vyhledávání, a proto se nabízí implementovat našeptávač. Existují knihovny, které obsahují hotové a plně funkční našeptávácí komponenty, nicméně moje zkušenost s nimi byla frustrující. Ano, komponenty prostě fungují, nicméně je nejde měnit. U každé knihovny, co jsem zkusil, jsem potřeboval upravit vzhled nebo chování, a to v některých případech šlo obtížně nebo vůbec. Tyto našeptáváče nesplňovaly má kritéria pro příjemné vyhledávání. Proto jsem se rozhodl napsat si vlastní našeptáváč. Takový, jaký bude příjemné používat.

2.2.1.2 adresář server

Obsahuje jeden soubor `api.js`. Ten definuje třídu `Client`, která obsahuje statické metody posílající `HTTP` požadavky na koncové body serverové části aplikace.

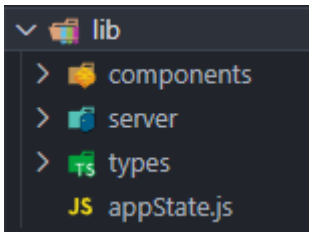
2.2.1.3 adresář types

Obsahuje javascriptové třídy. Ty zapouzdřují schéma a logiku dat přicházejících se serverovou částí aplikace. Například je zde definovaná třída `Card`, která zapouzdřuje data jednotlivých karet, také definuje atributy a metody pro jednoduchou manipulaci s kartami na úvodní stránce.

2.2.1.4 appState.js

Pokud je uživatel přihlášen, může vytvářet zájmy o karty. Toho docílí tak, že u karty vyhledané na úvodní stránce vybere, jestli chce kartu koupit nebo prodat. Pokud už o kartu zájem má, znovu vytvořit stejný zájem nejde, tlačítko není aktivní. Takto vytvořené zájmy může uživatel spravovat na stránce `/interests`. Data reprezentující uživatelovy zájmy musí mít jediný zdroj pravdy, přístupný ze všech stránek. K dosažení tohoto chování bylo použito `Svelte Store`.

- `Svelte Store` je knihovna pro správu stavu, která umožňuje sdílet a spravovat stav napříč komponentami. Funguje tak, že vytvoří jediný zdroj pravdy, ke kterému pak může přistupovat a aktualizovat ho jakákoli komponenta.
- `Store` je objekt, který má nějakou hodnotu a metody `subscribe`, `set` a `update`. Metoda `subscribe` bere jako argument metodu zpětného volání, která je zavolána pokaždé, když se hodnota `Store` změní.

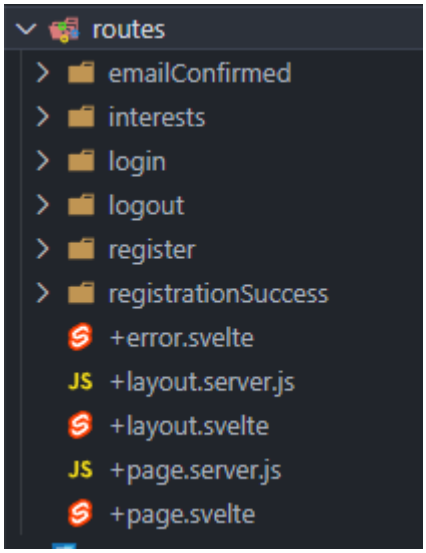


Obrázek 2: Adresář lib

2.2.2 Routes

Směrování je založeno na souborovém systému adresáře `routes`, který můžeme vidět na obrázku 3. Jednotlivé adresáře definují URL cesty. Každý adresář se skládá z více souborů, které jsou určeny jejich „+“ předponou. Například kořenová cesta „/“ se skládá z pěti souborů, které jsou vidět na obrázku 3.

- `+page.svelte`: komponenta definující stránku. Například `+page.svelte`, kterou lze vidět na obrázku 3, je úvodní stránka na URL adrese „/“.
- `+page.server.js`: definuje akce, které mohou nastat na dané adrese a funkci, která se vyhodnotí před načtením stránky. Například pro úvodní stránku potřebujeme před načtením načíst jména karet, aby pak mohl fungovat našeptávač.
- `+layout.svelte`: společná komponenta pro všechny stránky. Obsahuje například navigaci.
- `+layout.server.js`: může definovat akce a funkci, která se vyhodnotí před načtením. Akce a funkce jsou společné pro všechny stránky.
- `+error.svelte`: stránka, která se zobrazí v případě chyby.

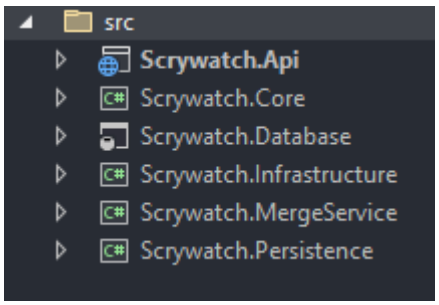


Obrázek 3: Adresář routes

2.3 Struktura serverové části projektu

Na obrázku 4 můžeme vidět strukturu serverové části projektu. Projekt je rozdělen do vrstev, které vychází z Clean architektury [5]. K propojení vrstev jsem použil dependency injection.

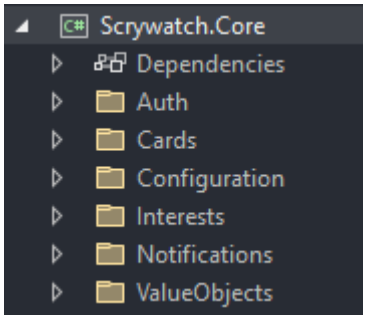
- Dependency injection (DI) je návrhový vzor používaný ke správě závislostí mezi různými částmi aplikace. Obecnou myšlenkou DI je vytvářet a spravovat objekty, které jsou používány jinými objekty, spíše než nechat každý objekt vytvářet a spravovat své vlastní závislosti.



Obrázek 4: Struktura serverové části projektu

2.3.1 Core

V této vrstvě jsou definovány entity a případy užití. Vertikalní strukturu vrstvy můžeme vidět na obrázku 5. Tato vrstva nemá závislost na žádné vrstvě. Jsou zde třídy a rozhraní, které definují doménu aplikace.



Obrázek 5: Struktura Core vrstvy

- `Auth`:
  - Registrovat se.
  - Podvrdit email.
  - Přihlásit se.
  - Odhlásit se.
- `Cards`:
  - Získat jména karet.
  - Získat karty se jménem.
- `Interests`:
  - Získat zájmy.
  - Vytvořit zájem.
  - Upravit zájem.
  - Smazat zájem.
- `Notifications`:
  - Poslat notifikaci emailem.

Při modelování jsem narazil na problém primitivní obsese [6], a to při modelování karet a jejich ceny.

- Primitivní obsese je anti-vzor, ke kterému dochází při nadměrném používání primitivních typů, zejména k modelování domény.

Cena karty závísí na měně a na tom, jaký má karta povrch. Pro reprezentaci povrchu i měny lze použít primitiv `string`. To ale vede k tomu, že v jiných vrstvách musíme provádět kontroly, jestli mají tyto primitiva validní stav. Lepší by bylo kdyby byly povrch karty i měna hodnotové objekty. To zajistí, že budou vždy ve správném stavu.

- Hodnotový objekt je jednoduchá entitia, jejíž rovnost nezáleží na identitě, ale na hodnotě.

```
1  public sealed record Card
2  {
3      public int Id { get; init; }
4
5      public string Name { get; init; }
6
7      public string Language { get; init; }
8
9      public string Rarity { get; init; }
10
11     public string CollectorNumber { get; init; }
12
13     public Set Set { get; init; }
14
15     public Face Face { get; init; }
16
17     public IEnumerable<Finish> Finishes { get; init; }
18
19     public IEnumerable<Price> Prices { get; init; }
20 }
21
22 public sealed record Price
23 {
24     public Finish Finish { get; init; }
25
26     public Currency Currency { get; init; }
27
28     public IDictionary<DateOnly, float> DateValuePairs { get; init; }
29 }
```

Zdrojový kód 1: Model karty a její ceny

2.3.2 Api

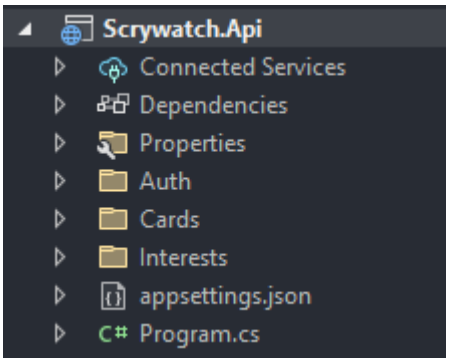
Na obrázku 6 můžeme vidět vertikalní strukturu api vrstvy. Tato vrstva definuje rozhraní pro komunikaci s okolním světem. Teoreticky by měla mít tato vrstva závislost pouze na Core vrstvě, to by ovšem vedlo k vytvoření další vrstvy, která bude mít jen účel poskládání všech vrstev do sebe. Rozhodl jsem se pro pragmatické řešení, kdy má vrstva Api závislost i na vrstvách Infrastructure a MergeService. Nicméně o jejich implementaci nic neví, při startu vytvoří hostovanou službu definovanou ve vrstvě MergeService a zavolá metodu `AddInfrastructure`, která se postará o spojení rozhraní definovaných ve vrstvě Core a jejich implementací ve vrstvě Infrastructure.

Pro implementaci API je použita třída `Controller`. Třída zahrnuje řadu akčních metod, což jsou veřejné metody, které jsou vyvolány MVC frameworkem v reakci na konkrétní požadavky.

- Model-View-Controller (MVC) framework je softwarový architektonický vzor, který rozděluje aplikaci na tři vzájemně propojené komponenty: mo-

del, view a controller.

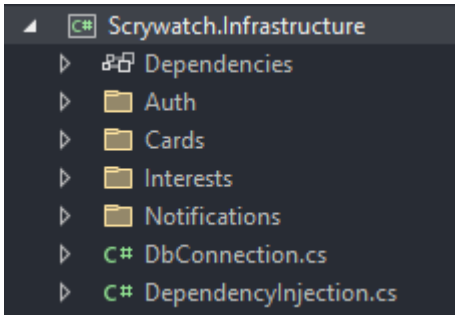
Každá metoda v `Controlleru` .NET je zodpovědná za zpracování konkrétního požadavku. K tomu používá rozhraní definovaná v Core vrstvě. `Controllery` v .NET také obvykle obsahují atributy, které definují, jak by měl MVC framework zpracovat požadavek a odpověď. Například atribut `[HttpGet]` označuje, že konkrétní metoda by měla zpracovávat požadavky `HTTP GET`, zatímco atribut `[HttpPost]` označuje, že by metoda měla zpracovávat požadavky `HTTP POST`.



Obrázek 6: Struktura Api vrstvy

2.3.3 Infrastructure

Na obrázku 7 můžeme vidět vertikální strukturu vrstvy Infrastructure. Tato vrstva implementuje funkcionalitu definovanou ve vrstvě Core.



Obrázek 7: Struktura Infrastructure vrstvy

Autentizace je implementována pomocí .NET Identity a JSON Web Tokenu.

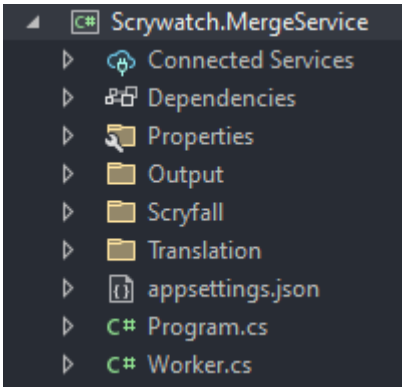
- .NET Identity je systém členství vestavěný do rámce ASP.NET, který poskytuje sadu funkcí pro správu ověřování uživatelů a autorizace v rámci webových aplikací. Umožňuje vývojářům snadno přidat do svých aplikací funkce registrace uživatelů, přihlášení a správy uživatelů a také implementovat funkce zabezpečení, jako je řízení přístupu na základě rolí a funkce resetování hesla.
- JSON Web Token (JWT) je otevřený standard pro bezpečný přenos informací mezi stranami jako objekt JSON. JWT se běžně používá pro účely ověřování a autorizace ve webových aplikacích, mobilních aplikacích a rozhraních API.

Komunikace s databází je implementována pomocí micro-ORM Dapper.

- Object-Relational-Mapping (ORM) je programovací technika, která umožňuje vývojářům mapovat data mezi relační databází a objektově orientovaným programovacím jazykem, jako je Java nebo C#. ORM poskytují sadu nástrojů a knihoven pro mapování databázových tabulek na třídy a databázových řádků na instance těchto tříd.
- Dapper je micro-ORM framework pro .NET, který poskytuje lehký a rychlý způsob mapování databázových záznamů na objekty .NET. Byl vyvinut týmem Stack overflow a vydán jako open-source projekt v roce 2011.

2.3.4 MergeService

Na obrázku 8 můžeme vidět strukturu vrstvy MergeService. Tato vrstva se stará o serializaci a synchronizaci s veřejnou API scryfall.com.



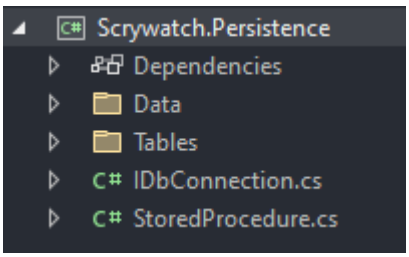
Obrázek 8: Struktura MergeService vrstvy

Scryfall.com jednou denně, v nepravidelný čas aktualizuje data. MergeService periodicky kotroluje datum změny těchto dat. Pokud se toto datum liší od data, kdy naposledy proběhla synchronizace, MergeService stáhne aktualizovaná data a předá je databázi. Pokud došlo ke změně dat, databáze je aktualizuje a vytvoří záznamy o všech cenách, které se změnily.

Potom, co tato synchronizace skončí, si MergeService od databáze vyžádá notifikace, které následně pošle emailem uživatelům.

2.3.5 Persistence

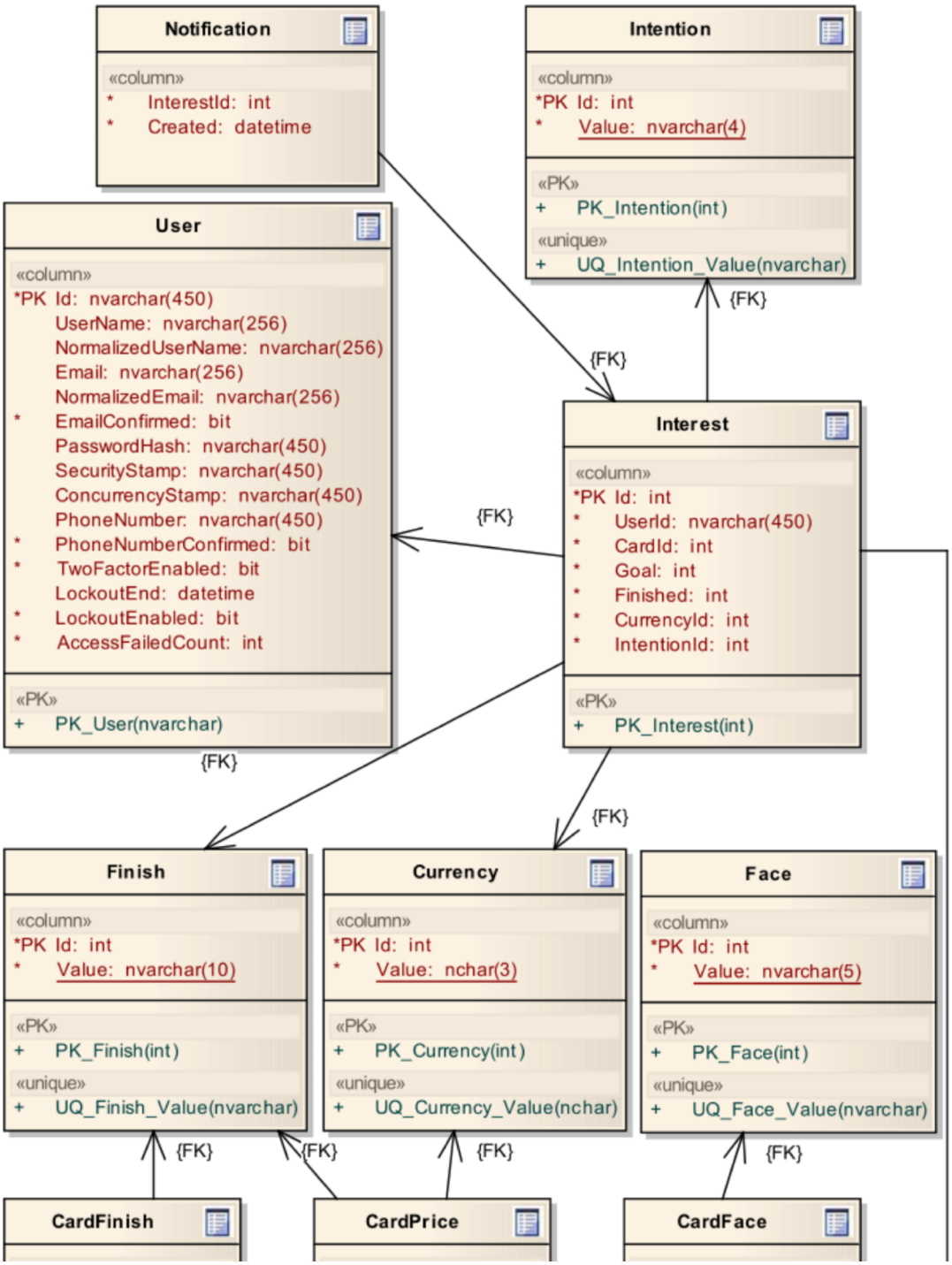
Na obrázku 9 můžeme vidět strukturu Persistance vrstvy. V této vrstvě jsou definovány třídy pro mapování dat z databáze, rozhraní pro komunikaci a statická třída `StoredProcedure`, která definuje názvy procedur uložených v databázi.



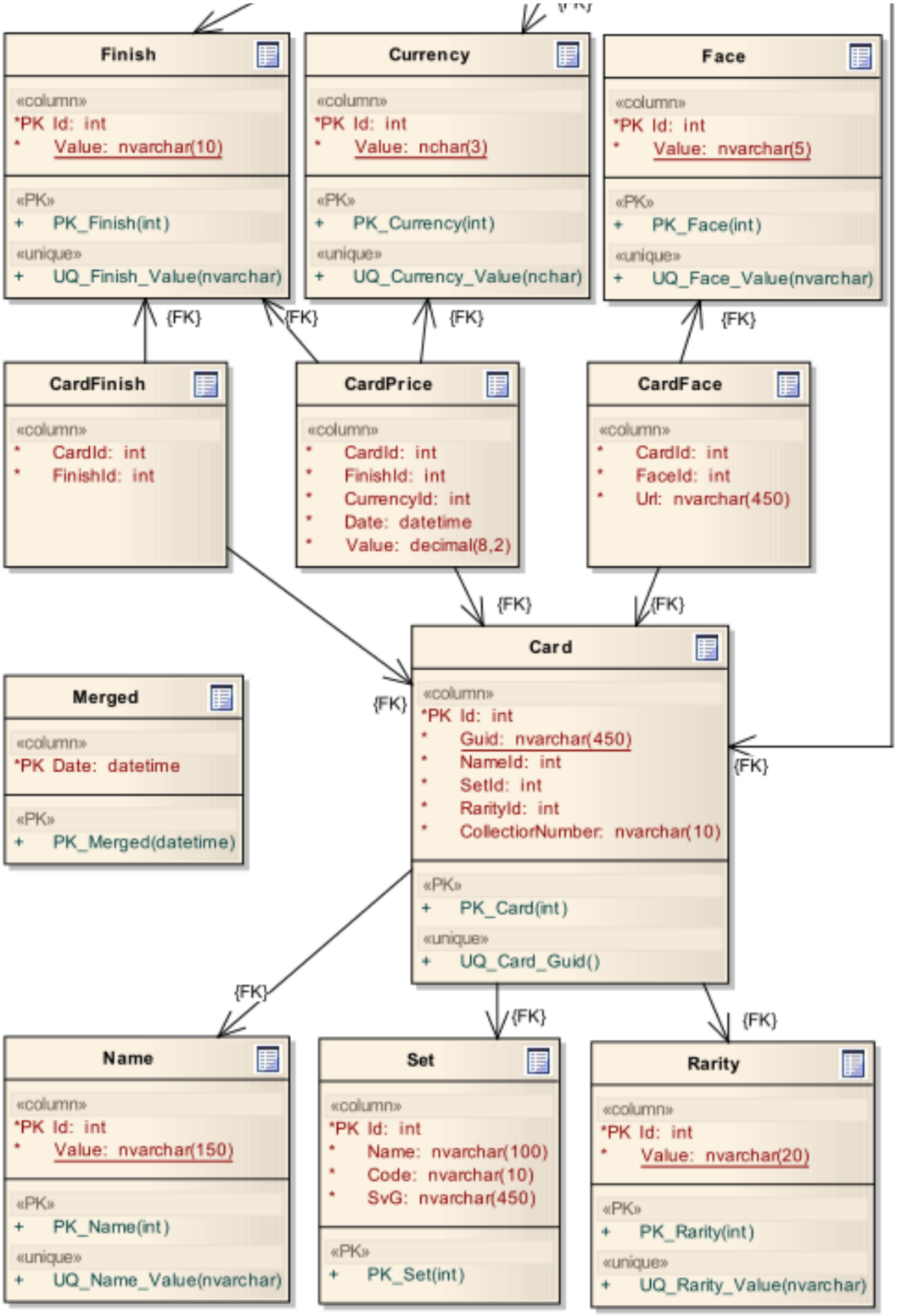
Obrázek 9: Struktura Persistence vrstvy

2.3.6 Database

Na obrázku 10 a 11 můžeme vidět diagram databáze.



Obrázek 10: Struktura Databáze



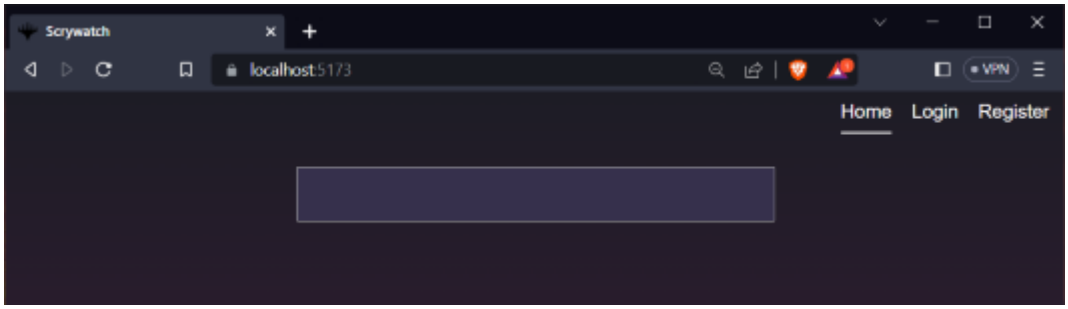
Obrázek 11: Struktura Databáze

3 Uživatelská dokumentace

Tato kapitola slouží jako manuál pro uživatele.

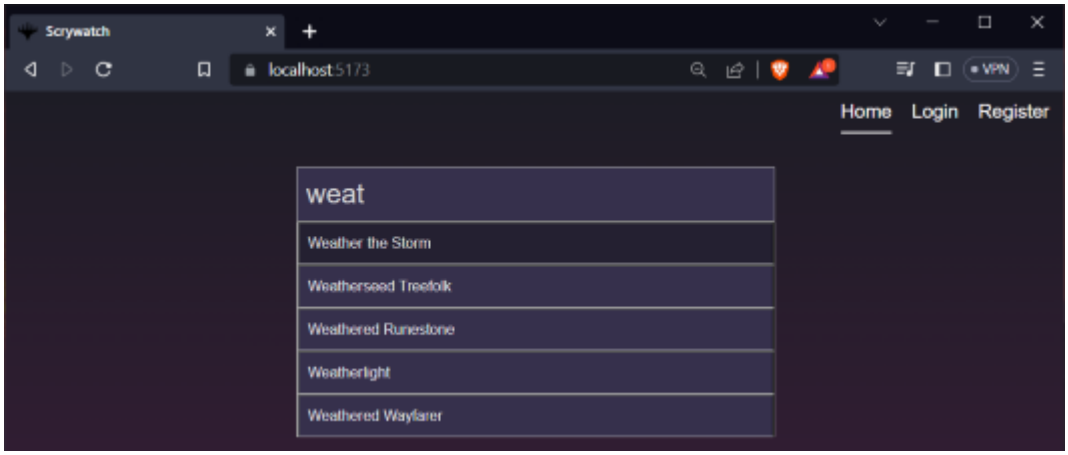
3.1 Vyhledání karty

Na úvodní stránce se nachází pouze navigace a vyhledávací pole.



Obrázek 12: Úvodní stránka

Pokud uživatel začne psát, zobrazí se našeptávač.



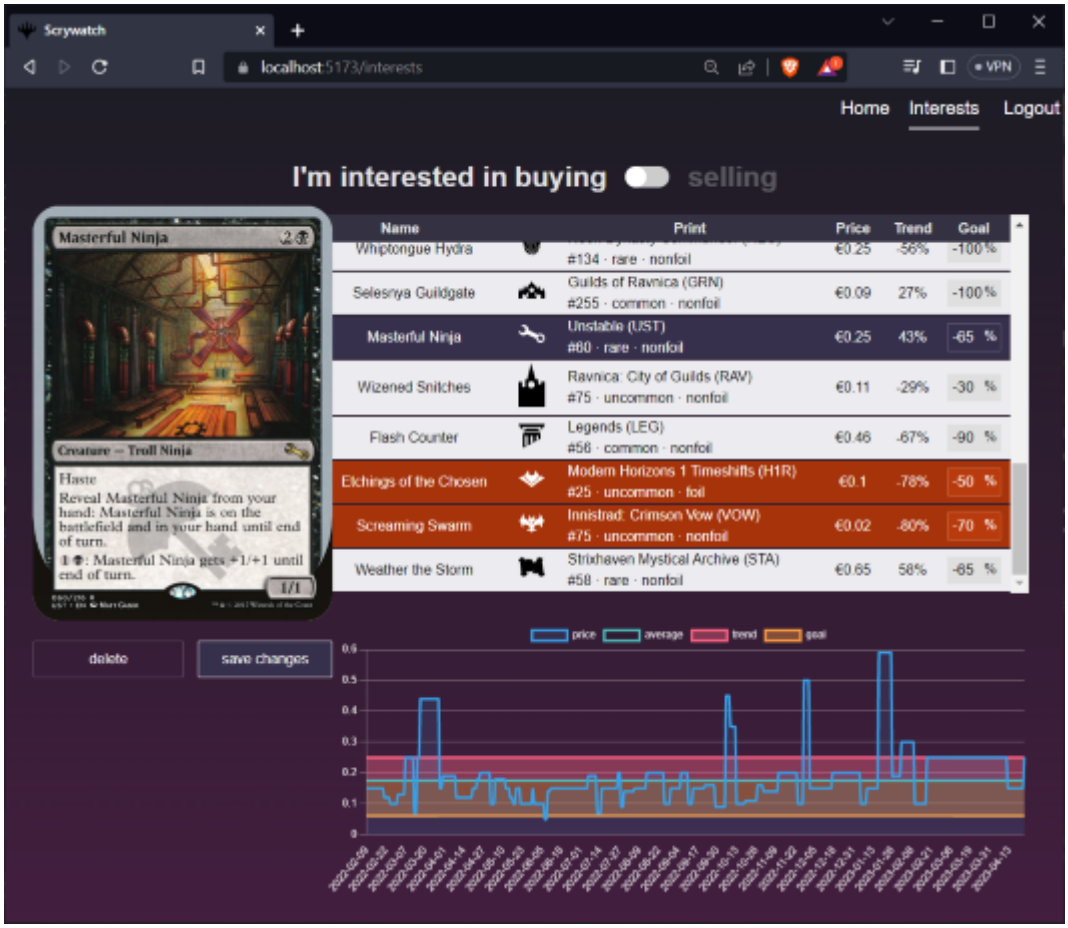
Obrázek 13: Úvodní stránka - našeptávání

Po vybrání jména se zobrazí obrázek karty, historie ceny a seznam všech karet s vybraným jménem. Cena závisí na povrchu karty a na měně. Pokud chce být uživatel notifikován na výkyv ceny u dané karty, musí si vytvořit účet, potvrdit emailovou adresu a přihlásit se.

Obrázek 14: Úvodní stránka - vybrané jméno

3.2 Nastavení notifikací

Uživatel si může nastavit na jak velký cenový výkyv chce být upozorněn. V seznamu jsou zvýrazněny karty, u kterých je výkyv splněn.



Obrázek 15: Zájmy

4 Rozšíření aplikace

Aplikace má mnoho potenciálních vylepšení, která jsem z časových důvodů vynechal. Rád bych je však v budoucnu implementoval.

4.1 Obnovení zapomenutého hesla

Obnovení zapomenutého hesla je zásadní funkcí, která by se musela dodělat před spuštěním aplikace pro veřejnost. Upřednostnil jsem práci na tom, jak aplikace vypadá.

4.2 OAuth

Dát uživateli možnost se neregistrovat a autentizovat přes Google, Facebook nebo jiný účet, jsem nepovažoval za zásadní. Je to ovšem něco, čeho si uživatelé cení, a chtěl bych ji implementovat v budoucnu.

4.3 Vypnout/zapnout emailové notifikace

Pokud uživatel už nechece dostávat emailové notifikace, musí se přihlásit do aplikace a smazat všechny své zájmy. To je poněkud nepraktické, a proto bych notifikace rozšířil o link, který by uživateli dal možnost zrušit posílání notifikací. Potom by to ovšem bylo potřebné dát uživateli možnost notifikace zapnout v případě, že jsou vypnuté.

4.4 Rozhraní pro nahrání existující kolekce

Archidekt.com a jiné stránky nabízí svým uživatelům možnost poskládat si balíček karet. Nahrání těchto karet by uživateli ušetřilo spoustu času, implementace ale bude pracná.

Závěr

Vyvinutá aplikace umožňuje pohodlně monitorovat ceny Magic: The Gathering karet. I neregistrovaný uživatel může vyhledávat karty a sledovat historii cen. Po registraci a potvrzení emailu se uživatel může přihlásit. Přihlášený uživatel může zvolit, jestli chce kartu koupit, nebo prodat, a vytvořit tak zájem. U každého zájmu lze nastavit při jaké výchylce bude poslána emailová notifikace.

Vytvořil jsem funkční responzivní webovou aplikaci, která zahrnuje intuitivní uživatelské rozhraní, funkční i na telefonech a webové API, které se synchronizuje s veřejným API scryfall.com a ukládá data o kartách do databáze.

Podařilo se mi splnit hlavní cíle práce a v budoucnu rád implementuji zmíněná rozšíření.

Conclusions

The developed application allows you to conveniently monitor the prices of Magic: The Gathering cards. Even an unregistered user can search for cards and see the price history. After registration and email confirmation, the user can log in. The logged-in user can choose whether he wants to buy or sell the card and thus create an interest. For each interest, it is possible to set when an email notification will be sent.

I've created a working responsive web app that includes an intuitive user interface that works on mobile devices and a web API that syncs with scryfall.com's public API and stores card data in a database.

I managed to fulfill the main objectives of the work and I am happy to implement the mentioned extensions in the future.

A Obsah přiloženého CD/DVD

Na samotném konci textu práce je uveden stručný popis obsahu přiloženého CD/DVD, tj. jeho závazné adresářové struktury, důležitých souborů apod.

bin/
: Obsahuje spustitelnou část serverové části aplikace pro zkopírování na webový server. Adresář obsahuje i všechny runtime knihovny a další soubory potřebné pro bezproblémový provoz webové aplikace na webovém serveru. Také obsahuje skript pro spuštění klientksé časti aplikace.

doc/
: Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

src/
: Kompletní zdrojové texty serverové a klientksé části programu se všemi potřebnými (příp. převzatými) zdrojovými texty, knihovnami a dalšími soubory potřebnými pro bezproblémové vytvoření spustitelných verzí programu / adresářové struktury pro zkopírování na webový server.

readme.txt
: Instrukce pro instalaci a spuštění aplikace, včetně všech požadavků pro jeho bezproblémový provoz.

data/
: Obsahuje databázi aplikace.

install/
: Instalátory aplikací, runtime knihoven a jiných souborů potřebných pro provoz programu , které nejsou standardní součástí operačního systému určeného pro běh programu.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.

Seznam zkratek

SSR Server Side Rendering

Literatura

[1] SVELTEKIT. *Introduction*. Dostupný z: ⟨https://kit.svelte.dev/docs/introduction⟩.

[2] *Svelte. : Cybernaticaly enhanced web apps*. Dostupný z: ⟨https://svelte.dev⟩.

[3] *React. : The library for web and native user interfaces*. Dostupný z: ⟨https://react.dev⟩.

[4] *Vue. : The Progressive JavaScript Framework*. Dostupný z: ⟨https://vuejs.org⟩.

[5] ROBERT, C. Martin. *The Clean Architecture*. Dostupný z: ⟨https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html⟩.

[6] *Primitive Obsession*. Dostupný z: ⟨https://refactoring.guru/smells/primitive-obsession⟩.