Gymnázium, Praha 6, Arabská 14

Programování

TÝMOVÝ ROČNÍKOVÝ PROJEKT

2023 **J.S., N.J., J.M.**

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

TÝMOVÝ ROČNÍKOVÝ PROJEKT

Předmět: Programování

Téma: Crypto Trading Platform

Autoři: Jan Sváček, Nikola Jankovič, Josef Mitošinka

Třída: 3.E

Školní rok: 2023/24

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Čestné prohlášení:

Prohlašujeme, že jsme jedinými autory tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené.

Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů udělujeme bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne ....

Anotace

Program, vytvořený v rámci této ročníkové práce, je zaměřený na problematiku výdělečnosti při obchodování s kryptoměnami. Uživatel má možnost vyzkoušet si svoje obchodní strategie nanečisto s pomocí Cryptonest. Veškeré transakce probíhající v naši ročníkové práci jsou pouze s 'papírovými' penězmi, tudíž se obchodníci nemusí bát o ztrátu finančních prostředků. Uživatel má možnost obchodvat s kryptoměnami v reálném čase s realnými kurzy. Také si může zobrazit grafy jednotlivých kryptoměn popřípadně si změti časová rozmezí grafů.

Abstract

The program created as part of this year's work is focused on the issue of profitability when trading cryptocurrencies. The user has the opportunity to try out their trading strategies with the use of Cryptonest. All transactions taking place in our annual work are using 'paper' money only, so traders do not have to worry about losing funds. The user has the option to trade cryptocurrencies in real-time with real rates. Traders can also view the charts of individual cryptocurrencies or even change the time frame of the charts.

Zadání ročníkového projektu

Cílem je vytvořit webovou aplikaci, která umožní uživateli obchodovat s kryptoměnami. Kurzy se budou v reálném čase updatovat. Uživatel bude moct nahlížet na grafy kurzů kryptoměn. Ačkoliv všechny hodnoty a grafy budou reálné, tak uživatel nebude obchodovat s vlastními reálnými penězi. Jednalo by se tedy o webovou aplikaci, která by působila jako opravdová tradovací aplikace, ale ovšem bez rizika finanční ztráty. To poskytne uživateli skvělou příležitost, si obchodování vyzkoušet.

Obsah

1 Úvod

1.1 Obchodování s kryptoměnami

V kontextu stále se rozvíjející oblasti kryptoměn se naše webová aplikace zabývá problematikou výdělečnosti při obchodování s těmito digitálními měnami. Náš projekt pojmenovaný Cryptonest, vznikl s cílem poskytnout uživatelům prostředí pro zkoumání a testování strategií pro obchodování s kryptoměnami v bezpečném virtuálním prostředí bez rizika finanční ztráty.

Naše aplikace přináší uživatelům možnost sledovat aktuální kurzy kryptoměn v reálném čase a umožňuje jim investovat virtuální peníze do kryptoměny nebo kryptoměn dle jejich výběru. Tímto způsobem uživatelé získávají cenné informace a přehled, které jim pomáhají lépe porozumět dynamice kryptoměnových trhů.

1.2 Struktura

Projekt se skládá ze tří částí. První částí je frontend, tedy část která se zobrazuje uživateli. V našem případě to jsou to například grafy, nabídky kryptoměn a přehled portfolia uživatele.

Druhou částí je backend. To je program běžící na serveru, který se uživateli nezobrazí ale komunikuje s frontendem a umožňuje zobrazení správných a přesných dat. Stará se o autentifikaci uživatelů a sbírání dat pro vytváření grafů a dalších částí aplikace.

Poslední část je databáze která ukládá data o uživatelích, obchodech a hodnotách, které jsou důležité pro správnou zákaznickou zkušenost.

2 Řešení problematiky

Jedním z hlavních problémů mnohých obchodníků s kryptoměnami je výdělečnost. Najít dobrou strategii, která vám je schopná dlouhodobě vydělávat finance je velmi obtížné, a kvůli tomu je pouze jedno procento všech obchodníků na burzách výdělečné. Proto jsme vytvořili aplikaci s názvem Cryptonest. Jedná se o paper trading aplikaci, která umožňuje uživatelům kupovat a prodávat kryptoměny v reálném čase bez rizika na finančních ztrátách.

Po zaregistrování či přihlášení se do aplikace si uživatel může vyhledat libovolnou kryptoměnu a dále se mu otevře možnost s ni obchodovat. Uživatel má možnost sledovat historický graf kryptoměny, na kterém je možné měnit časový rámec dle potřeby. Dále se může podívat na počet vytěžených tokenů či na jejich maximalní možný počet, jelikož většina kryptoměn má omezené maximalní množství tokenů. Jedna z dalších užitečných informací, jenž má uživatel možnost vidět, může být procentuální změna ceny kryptoměny za posledních 24 hodin nebo její průměrná cena. Aby uživatel nemusel neustále hledat jednu a tu samou kryptoměnu, kdykoliv se chce podívat na její cenu či provést transakci, tak má možnost si ji přidat do oblíbených, kde ji následně uvidí hned po přihlášení do aplikace na hlavní stránce.

3 Použité technologie

Začněme backendovou částí. Ta je postavena na frameworku Laravel, který je jedním z nejpopulárnějších a nejlepších frameworku pro vývoj webových aplikací v jazyce PHP. Laravel využívá MVC architektury (Model-View-Controller), která nám umožňuje logické oddělení datové vrstvy, prezentační vrstvy a aplikační logiky. Díky tomu máme přehledný a dobře strukturovaný kód.

Pro komunikaci s naší MySQL databází využíváme ORM (Object-Relational Mapping). Ten nám dovoluje pracovat s databázi pomocí objektů a tříd, díky čemuž je manipulace s daty jednodušší.

Pro správu MySQL databáze využíváme Docker, který nám umožní vytvořit izolované kontejnery pro naši aplikaci. Kromě toho používáme phpMyAdmin, který nám poskytuje dobré, uživatelsky přívětivé rozhraní pro práci s MySQL databází.

Na frontendové časti jsme se rozhodli použít framework Vue.js. To nám umožňuje vytvořit znovupoužitelné komponenty jako jsou tlačíka či grafy kryptoměn, které pak můžene jednoduše importovat do naše kódu.

Pro stylování stranek jsme místo klasického CSS zvolili Tailwind CSS. Pomocí Tailwind CSS je kód přehlednější a rychleji se píše. Nemusíme kombinovat HTML a CSS soubory a styly můžeme definovat přímo v HTML.

4 Hlavní funkce webové aplikace

4.1 Dashboard

Hlavní stránka naší aplikace se nachází na root url adrese (/). Na této stránce se nachází základní informace o naší aplikaci, tabulka populárních kryptoměn, register a login nebo také sekce Frequently Asked Questions. Poté co se uživatel zaregistruje a přihlásí, se může dostat na /dashboard, kde se mu zobrazí základní rozhraní verifikovaného uživatele. K tomu složí route::get dashboard. Ta prvně zkontroluje, zda je uživatel opravdu přihlášený a následně zavolá Controller a jeho třídu renderDashboard, kde se nachází logika pro vyrenderování view.

```
1 Route::get('/dashboard', [\App\Http\Controllers\DashboardController::class, '
    renderDashboard2'])->middleware(['auth', 'verified'])->name('dashboard');
```

Listing 1: Route get dashboard

V controlleru si do proměnné na řádku 2 uložíme oblíbené kryptoměny uživatele. Metoda getFavouriteCryptoCurrencies získá prostřednictvím ORM jména oblíbených kryptoměn uživatele z databáze. Ve foreach loopu se ke každé kryptoměně získají z burzy konkrétní data. K tomu slouží metoda createObject, která dostane jako parametr jméno kryptoměny.

Nakonec pomocí Inertia::render vrátíme uživateli view - Dashboard.vue s propy, díky kterým posílame data z backendu do frontendu.

```
1  protected function renderDashboard2() {
2          $favouriteCryptoCurrencies = $this->getFavouriteCryptoCurrencies();
3          $ListOfCurrencies = [];
4          foreach ($favouriteCryptoCurrencies as $key => $favouriteCryptoCurrency
       ) {
5              $ListOfCurrencies[] = $this->createObject($favouriteCryptoCurrency-
       >name);
6          }
7          return Inertia::render('Dashboard', [
8              'ListOfCurrencies' => $ListOfCurrencies,
9              'error_message' => '',
10         ]);
11 }
```

Listing 2: DashboardController

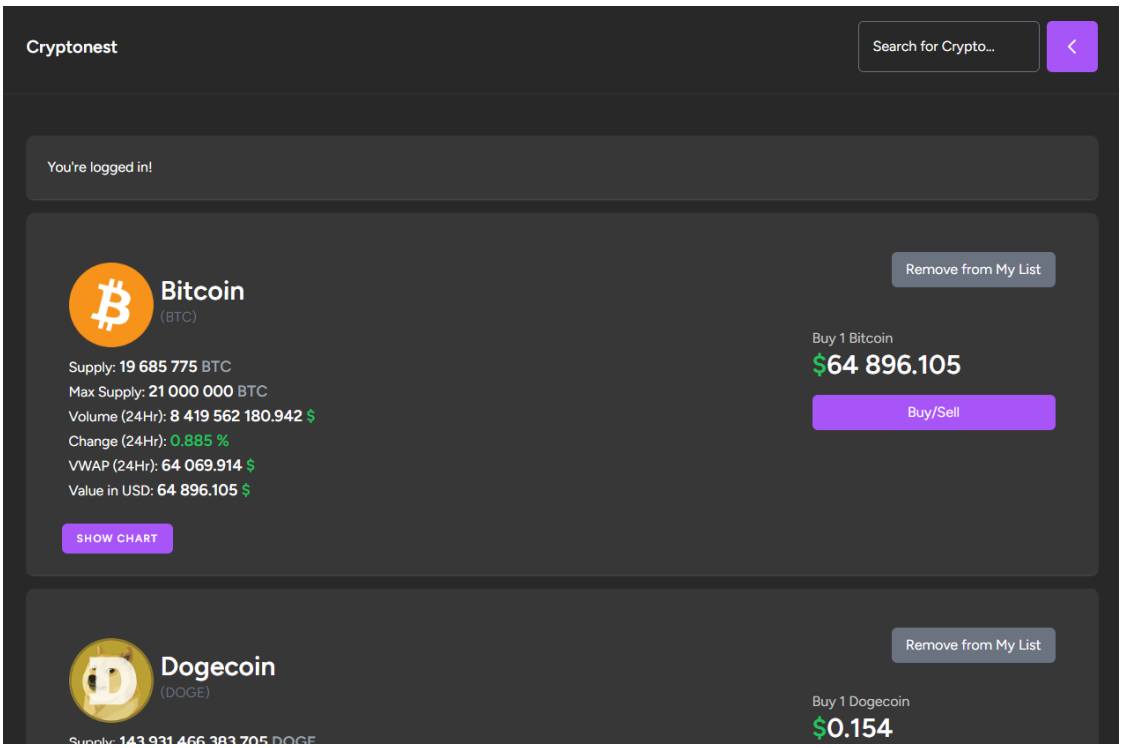
Zde posíláme oblíbené kryptoměny a jejich hodnoty.

Ve frontendové části si nějak musíme poradit s daty. Pro to jsou klíčové dva komponenty - CryptoCurrency.vue a CryptoChart.vue. Oba dva komponenty přijímaji data, které následně přehledně uživateli zobrazí. Komponenty se generují pro každou kryptoměnu, kterou si uživatel přidal do olíbených a proto jsme použili v-for, což je vlastně foreach loop ve vue.js.

```
<span v-if="ListOfCurrencies">
    <div v-for="currency in ListOfCurrencies" class="bg-bg mb-4 overflow-hidden
     shadow-sm shadow-primarytext/20 sm:rounded-lg">
        <div class="p-6 text-primarytext bg-secondarybg">
            <CryptoCurrency
            :cryptocurrencyName="currency['values']['data']['name']"
            :shortcut="currency['values']['data']['symbol']"
            :priceUsd="currency['values']['data']['priceUsd']"
            :supply="currency['values']['data']['supply']"
            :maxSupply="currency['values']['data']['maxSupply']"
            :volumeUsd24Hr="currency['values']['data']['volumeUsd24Hr']"
            :changePercent24Hr="currency['values']['data']['changePercent24Hr']
    "
            :vwap24Hr="currency['values']['data']['vwap24Hr']"
            :currencyImg="'https://cryptologos.cc/logos/' + currency['values
    ']['data']['id'] + '-' + currency['values']['data']['symbol'].toLowerCase()
     + '-logo.png?v=029'"
            :addedToList=true />
            <PrimaryButton @click="toggleVisibility(currency)" class="mx-4">
                {{ currency.showChart ? 'HIDE Chart' : 'SHOW Chart' }}
            </PrimaryButton>
            <CryptoChart v-if="currency.showChart"
            :dataIn24="currency['data24']"
            :dataIn12="currency['data12']"
            :dataIn1="currency['data1']"
            :name="currency['name']" />
        </div>
    </div>
</span>
```

Listing 3: Dashboard.vue

Graf hodnoty kryptoměny za časový úsek se zobrazí nebo schová pomocí tlačítka show nebo hide chart. V základním nastavení je při zapnutí stánky graf schovaný.



Dashboard vypadá následovně.

4.2 Vyhledávání kryptoměny

Uživatel si na stránce dashboard může vyhledávat nejrůznější kryptoměny. K tomu nám slouží dynamická url adresa.

```
Route::get('/dashboard/{cryptocurrency}', [\App\Http\Controllers\
    CryptoDetailController::class, 'singleCryptoCurrency'])->middleware(['auth'
    , 'verified'])->name('dashboard-search');
```

Listing 4: Route dashboard dynamická url

Když si uživatel vyhledá kryptoměnu přesune ho aplikace na url dashboard/{cryptocurrency}. Parametr cryptocurrency závisí na tom, co uživatel do vyhledávacího pole zadal. Díky tomu můžeme v controlleru jednoduše zjisit zda kryptoměna, kterou se uživatel snaží najít, existuje a pokud ano, tak ji zobrazit. Údaje o kryptoměně získáme pomocí getSingleAsset a data potřebná pro graf získame pomocí getAssetHistory. $interval udává časový úsek dvou bodů na grafu např 'm15' znamená, že bod na grafu bude uvádět hodnotu, kterou kryptoměna nabyla po 15 minutách od posledního bodu.

```
Cryptocap::getSingleAsset($cryptocurrency);
Cryptocap::getAssetHistory($cryptocurrency, $interval);
```

Listing 5: Cryptocap package

Všechny tyto data pak příjmou již zmíněné komponenty - CryptoCurrency.vue a CryptoChart.vue a to se pak přehledně zobrazí uživateli.

4.3 Nákup a Prodej kryptoměn

V komponentu CryptoCurrency.vue je tlačítko na nákup kryptoměny. Poté co na něj uživatel klikne, tak se přesune na url adresu dashboard/{cryptocurrency}/buy. Za pomocí dynamické url adresy se nám dobře pracuje s požadavkem. V controlleru zase získáme potřebné data o kryptoměně a pošleme je do frontendu, konkrétně do souboru Purchase.vue.

```
Route::get('dashboard/{cryptocurrency}/buy', [BuyCryptocurrencyController::
    class, 'displayPurchase'])->middleware(['auth', 'verified'])->name('
    buyCryptocurrency');
```

Listing 6: Route pro koupi krypotměny

Na této stránce je klíčový komponent CryptoCalculator.vue, který umožňuje uživateli buď kryptoměnu prodat nebo koupit. Když chce uživatel kryptoměnu koupit, do horního pole zadá, kolik dolarů chce investovat a v dolním poli se mu ukazuje, kolik kryptoměny si za to koupí. Když chce kryptoměnu prodat funguje to stejně ale naopak. Tudíž zadá množství kryptoměny, kterou chce prodat a zobrazí se mu hodnota tohoto množství v dolarech. Poté, co uživatel zadal potřebné údaje ke koupi/prodeji klikne na tlačitko buy/sell.

```
<ToggleButton @click="buyCrypto" v-if="mode === 'buy'" :textBefore="'Buy Crypto
    '" :textAfter="'Bought Successfully'"/>
<ToggleButton @click="sellCrypto" v-else :textBefore="'Sell Crypto'" :textAfter
    ="'Sold Successfully'"/>
```

Listing 7: Toggle button buy/sell

```
async buyCrypto() {
  const response = await axios.post('/buy-crypto', {
    cryptocurrency: this.$props.cryptocurrency.id,
    cryptoAmount: parseFloat(this.cryptoAmount),
    usdAmount: parseFloat(this.usdAmount)
  });
},
async sellCrypto() {
  const response = await axios.post('/sell-crypto', {
    cryptocurrency: this.$props.cryptocurrency.id,
    cryptoAmount: parseFloat(this.cryptoAmount),
```

```
    usdAmount: parseFloat(this.usdAmount)
  });
},
```

Listing 8: Metody pro koupi/prodej kryptoměny

Kliknutí na tlačítko spustí s ním vázanou funkci. Ta pošle request s těmito údaji: jméno kryptoměny, počet dolarů a počet kryptoměny. K tomu pak odpovídá příslušná route, která zavolá odpovídající metodu controlleru.

```
Route::post('/buy-crypto', [BuyCryptocurrencyController::class, 'buyCrypto']);
Route::post('/sell-crypto', [BuyCryptocurrencyController::class, 'sellCrypto'])
```

Listing 9: Post routes

Tentokrát se nejedná o get route ale o post route. V controlleru musíme provést zápis do databáze. Prvně musíme změnit, kolik má uživatel peněz. Poté změnit, kolik má uživatel kryptoměny, kterou koupil nebo prodal. Následně musíme vytvořit záznam o tomto tradu.

Musíme samozřejmě zkontrolovat, zda má uživatel dostatek prostředků na tento trade. Zápis tradu do tabulky trades pomocí ORM vypadá takto.

```
function saveTrade($userId, $cryptocurrency, $cryptoAmount, $usdAmount, $
    boughtCrypto) {
    Trade::create([
        'user_id' => $userId,
        'name' => $cryptocurrency,
        'crypto_amount' => $cryptoAmount,
        'usd_amount' => $usdAmount,
        'bought_crypto' => $boughtCrypto,
    ]);
}
```

Listing 10: Create Trade

U peněz a množství kryptoměny nejprve z databáze získáme tato data a pak k ním buď přidáme nebo odebereme hodnoty, které jsme od uživatele dostaly.

4.4 Balance

Poslední hlavní duležitou funkcí naší aplikace je rozhraní, kde uživatel může sledovat všechny data, které jsou pro něj důležité. V příslušném controlleru získáváme stav konta uživatele, všechny kryptoměny, které vlastní a kolik jich má, jeho oblíbené kryptoměny a historii všech tradů, které uživatel udělal. Kromě toho, že uživatel vidí, kolik má jaké kryptoměny, přepočítáváme u každé kryptoměny ekvivalent v dolarech. Také počítáme, kolik má uživatel celkově peněz v kryptoměnách a také kalkulujeme portfolio uživatele, kde sčítáme kolik má dolarů u nás ve virtuální peněžence a jakou hodnotu mají jím koupené kryptoměny.

Oblíbené kryptoměny jsou zároveň i proklikávací odkazy, aby uživatel mohl snadno sledovat grafy a detail kryptoměny, která ho zajímá.

5 Řešení klíčových problémů

5.1 Získávání dat z burzy

Na získávání dat z burzy využíváme package Cryptocap pro Laravel. Cryptocap propojuje naši aplikaci s veřejným API poskytovatele CoinCap, který nám umožňuje přístup k veškerým potřebným informacím o danné kryptoměně, jako jsou aktuální cena či tržní kapitalizace. CoinCap sbírá informace z burz jako jsou Binance, CoinBase Pro nebo Bitfinex, tudíž ho je možné považovat za spolehlivý zdroj dat.

Pro získání údajů o kryptoměně používáme metodu

```
Cryptocap::getSingleAsset('název kryptoměny');
```

Listing 11: CryptoInfo

. Tato metoda nám umožní získat data o aktuální cenně, objemu obchodů, tržní kapitalizaci a dalších důležitých parametrech.

Na získání historických dat o kryptoměně využíváme metodu 'Cryptocap::getAssetHistory (název kryptoměny, časový interval)', kterou jsme z části udělali sami, jelikož autor Cryptocapu nevytvořil parametr pro zvolení časového úseku.

K vytvoření grafu jsme využili knihovy Chart.js. Tato knihovna nám umožňuje vizualizaci historických dat ve formě grafu, který uživateli poskytne lepší přehled o historii ceny kryptoměny.

5.2 Tvoření Grafů

Pro tvoření grafů jsme se rozhodli použít knihovnu jménem chart.js. Jedná se o jednoduchou, flexibilní a moderní javascriptovou knihovnu. Protože je graf klíčový pro obchod s kryptoměnami, rozhodli jsme se vytvořit komponent CryptoChart.vue. To nám umožní graf zobrazovat jednodušše na různých místech webové aplikace.

Komponent dostane více sad dat. Data za poslední měsíc, kde e interval má jeden den. Data za poslední týden, kde interval má 2 hodiny. Dále data za poslední den, půl dne, 6 hodin a poslední hodinu s tím, že se intervaly stále zkracují. Tímto způsobem získame z burzy data za pomocí Cryptocap package.

```php
$cryptoDataIn1M = $this->getHistoryData($cryptocurrency, 'd1', 720);
$cryptoDataIn1W = $this->getHistoryData($cryptocurrency, 'h2', 168);
$cryptoDataIn24 = $this->getHistoryData($cryptocurrency, 'h1', 25);
$cryptoDataIn12 = $this->getHistoryData($cryptocurrency, 'm30', 12.5);
$cryptoDataIn6 = $this->getHistoryData($cryptocurrency, 'm15', 6);
$cryptoDataIn1 = $this->getHistoryData($cryptocurrency, 'm1', 1);
```

Listing 12: Získávání dat do grafu

Funkci getHistoryData jsme si vytvořili sami a voláme ji protože z dat, které dostaneme musíme vyfiltrovat jen ty, které chceme. Časový úsek určuje poslední parametr funkce v hodinách.

```php
protected function getHistoryData($cryptocurrency, $interval, $time_period) {
    $rawHistoryData = json_decode(json_encode(Cryptocap::getAssetHistory($cryptocurrency, $interval)), true);
    $filteredHistoryData = $this->filterDataForTimeInterval($rawHistoryData, $time_period);

    return $filteredHistoryData;
}
```

Listing 13: Funkce get history data

```php
protected function filterDataForTimeInterval($data, $intervalInHours) {
    $intervalInMilliseconds = $intervalInHours * 60 * 60 * 1000;

    $currentTimestampMs = round(microtime(true) * 1000);
    $startTimeMs = $currentTimestampMs - $intervalInMilliseconds;

    $filteredData = array_filter($data['data'], function ($item) use ($startTimeMs, $currentTimestampMs) {
        return isset($item['time']) && $item['time'] >= $startTimeMs && $item['time'] <= $currentTimestampMs;
    });

    // Reindex the array to start from 0
    $filteredData = array_values($filteredData);

    return ['data' => $filteredData];
}
```

Listing 14: Funkce filter data

Do proměnné $rawHistoryData uložíme historické data kryptoměny z burzy. Díky tomu dostaneme historická data rozdělená na správné intervaly. Nedostaneme ovšem námi zadány časový úsek a proto musíme ještě data filtrovat ve funckci filerDataForTimeInterval, která dostane hrubá data a časový úsek v hodinách, který požadujeme. Prvně je třeba převést hodiny na milisekundy, protože v hrubých datech je u každé hodnoty uveden UNIX čas v milisekundách. Následně zjistíme aktuální čas a pak filtrujeme data tak, že požadujeme aby hodnoty byly od začátku uživatelem zadaného intervalu až do aktualního času. Poté pole přeindexujeme pro snadnější použití později a toto pole vrátíme.

Tyto data pak pošleme do frontendu, kde je dostane již zmíněný komponent CryptoChart.vue. Ten ke každé sadě dat vytvoří jeden graf. Pomocí komponentu DropdownInterals.vue se dá mezi sady dat jednodušše volit. Tento komponent pak emitne interval, na který uživatel klikl a CryptoChart komponent zobrazí odpovídající graf. Takto graf dostane data, která zobrazí.

```
const data = {
    labels: labels,
    datasets: [
      {
        label: 'Price (USD) X ' + props.name,
        backgroundColor: backgroundColor,
        borderColor: borderColor,
        data: selectedData.data.map((entry) => parseFloat(entry.priceUsd)),
      },
    ],
  };;
```

Listing 15: Dataset grafu

Když komponent DropdownIntervals emitne změnu v intervalu, CryptoChart změní selectedData na data, která odpovídají tomu, na co uživatel kliknul. Kromě dat také změní barvu grafu.

```
<template>
    <div>
        <label for="interval" class="text-primarytext text-lg mb-2 mr-2">Select
     Interval:</label>
        <select v-model="selectedInterval" @change="handleChange" class="p-2
    border-2 rounded bg-purple-500 text-white cursor-pointer focus:border-
    purple-500 active:border-purple-500 focus:ring focus:ring-purple-300">
            <option value="M">Last Month</option>
            <option value="W">Last Week</option>
            <option value="24">1 Day</option>
            <option value="12">12 Hours</option>
            <option value="6">6 Hours</option>
            <option value="1">1 Hour</option>
        </select>
    </div>
</template>

<script setup>
import { ref, defineEmits } from 'vue';
const emit = defineEmits();
const selectedInterval = ref('24');
const handleChange = () => {
    emit('update:selectedInterval', selectedInterval.value);
};
</script>
```

Listing 16: DropdownIntervals

```
  if (selectedInterval.value === '12') {
    selectedData = props.dataIn12;
    backgroundColor = 'rgba(255, 0, 0, 0.2)';
    borderColor = 'rgb(255, 0, 0)';
  } else if (selectedInterval.value === '1') {
    selectedData = props.dataIn1;
    backgroundColor = 'rgba(255, 255, 0, 0.2)';
    borderColor = 'rgb(255, 255, 0)';
  } apod. pro další intervaly
```

Listing 17: Měnění dat a barvy

Zde můžeme vidět příklad grafu.

5.3 Responzivita

Responzivita webových stránek je schopnost stránek přizpůsobit se různým zařízením a obrazovkám, jako jsou počítače, tablety a mobilní telefony. Cílem responzivního designu je zajistit, aby uživatelé mohli snadno a efektivně používat webové stránky bez ohledu na velikost jejich obrazovky nebo zařízení, které používají.

Abychom tohoto docílili, využíváme už předem zmíněný framework Tailwind CSS. Tento framework nám umožňuje psát jednotlivé css styly přímo do html kódu jako předem definované třídy (například třída p-4 přidá objektu padding velký 4 pixely). Aby byla stránka responzivní, využívámě takzvaných breakpointů implementovaných v tailwindu. Breakpointy jsou třídy, které se používají k definici responzivních layoutů a stylů na základě šířky obrazovky. Například, pokud chcete aby se určitý prvek choval jinak na zařízení s šířkou 640px a vyšší využili byste breakpointu 'sm:'. Na příkladu vidíme, že na zařízeních s velikostí okna menší než 640px bude tento sloupec tabulky hidden a od 640px výše se bude zobrazovat s display hodnotou table-cell.

```
1    <th scope="col" class="px-6 py-3 hidden sm:table-cell">
2        Shortcut
```

```
    </th>
```

Listing 18: Příklad využití breakpointu

5.4 Dark/Light módy

Jednou z dalších funkcionalit Tailwind CSS je jednoduché implementování tmavého režimu. Dark mode je populární funkce, kterou má skoro každá moderní webová aplikace. Tmavý režim umožnuje uživatelům změnit barevné spektrum aplikace na očím příjemnější tmavou barvu. V naší aplikaci tohoto dosáhneme pomocí definování barev předem definovaných v souboru app.css, které poté používáme v celém kódu. V tomto souboru jsou nadefinované tři módy: Light, Dark a High Contrast. Tyto módy poté mají stejně pojmenované barvy, ale v každém módu mají jiné hodnoty pro jednoduché přebarvení celé stránky po jediném kliknutí od uživatele v sekci profile. Zároveň nám tailwind umožňuje zjistit předvolbu systému, který uživatel používá, a pokud má své zařízení nastavené v tmavém režimu, naše stránka bude automaticky také v tmavém režimu.

5.5 Loga kryptoměn

Při vyvíjení designu webové aplikace jsme chtěli, aby jsme spolu s grafem a ostanímy daty kryptoměny zobrazili i její logo. Jelikož kryptoměn, se kterými může uživatel u nás obchodovat je opravdu mnoho, tak manuální stažení obrázku nepřipadalo v úvahu. Naštěstí jsme našli stránku, kde jsou loga všech kryptoměn a url adresy jednolivých log krypotměn jsou vytvořena tak, že se dají dynamicky volat v naší aplikaci. Například logo bitcoinu je na url adrese https://cryptologos.cc/logos/bitcoin-btc-logo.png?v=031 nebo například logo dogecoinu je na url adrese https://cryptologos.cc/logos/dogecoin-doge-logo.png?v=031. Díky tomu, že se adresy až na název a zkratku schodují, jsme schopni získat odkaz na jakékoliv logo kryptoměny bez problémů.

```
<img :src="'https://cryptologos.cc/logos/' + cryptocurrency['name'].toLowerCase
    () + '-' + cryptocurrency['symbol'].toLowerCase() + '-logo.png?v=029'"/>
```

Listing 19: Crypto Logo

6 Datové struktury

6.1 Vue.js a komponenty

Vue.js je JavaScryptový framework určený k vývoji uživatelských rozhraní a jedná se o jednoduchý a flexibilní nástroj pro tvorbu interaktivních webových aplikací.

Komponenty jsou ve Vue.js samostatné, znovupoužitelné bloky kódu. To nám umožňuje strukturovat uživatelské rozhraní aplikace do logických a nezávislých částí, což usnadňuje správu a údržbu kódu. Při vytvoření komponentu definujete její šablonu pomocí template tagu. Do tohoto tagu napíšete veškerý HTML kód a v našem případě i stylizujete pomocí tříd Tailwind CSS. Poté můžete případně přidat nějaký JavaScriptový kód pomocí tagu script.

Příkladem takového komponentu je v našem kódu například FAQ.vue. Tento komponent prvně definuje proměnné question a answer a poté v odděleném script tagu metodu toggleVisibility, která umožňuje po zmáčknutí tlačítka zobrazit odpověď. Následuje tag template, kde už pouze napíšeme html kód, stylizujeme a zavoláme proměnné question a answer. Tento komponent poté voláme v Index.vue, kde vyplníme tyto textové proměnné otázkou a odpovědí. Toto nám umožňuje jednoduchou znovupoužitelnost kódu a není nutné zbytečné opakování.

```
<script setup>
defineProps({
  question: String,
  answer: String,
})
</script>

<script>
export default {
  data() {
    return {
      isVisible: false
    };
  },
  methods: {
    toggleVisibility() {
      this.isVisible = !this.isVisible;
    }
  }
```

```
}
</script>

<template>
  <div class="p-6 bg-secondarybg flex flex-col md:justify-between my-2 rounded-
    md w-full shadow-md">
    <div class="text-primarytext relative">
      <div class="flex items-center justify-between">
        <div class="flex items-center w-full">
          <p class="text-xl font-bold inline-block mr-2">
            {{ question }}
          </p>
          <button @click="toggleVisibility"
            class="bg-inherit rounded-full p-2 inline-flex items-center justify
    -center focus:bg-buttonbg font-bold text-xl ml-auto transition-all"
            style="width: 2.5rem; height: 2.5rem;">
            {{ isVisible ? '-' : '+' }}
          </button>
        </div>
      </div>
      <div v-if="isVisible" class="p-2 mt-4">
        <p class="text-lg text-secondarytext block ml-2">
          {{ answer }}
        </p>
      </div>
    </div>
  </div>
</template>
```

Listing 20: FAQ.vue komponent

```
    <FAQ question="Why choose Cryptonest?"
        answer="Cryptonest offers a risk-free enviroment, which is suitable
     even for complete beginners." />
```

Listing 21: Příklad volání komponentu FAQ.vue

6.2 ORM & Databáze

6.2.1 Modely

V Laravelu existují tzv. modely, což jsou třídy reprezentující jednotlivé tabulky v databázi. Jeden model odpovídá jedné tabulka. Modely nám usnadňují manipulování s daty v tabulkách pomocí objektivně orientovaného přístupu. Každý model pak obsahuje nějaké proměnné, které odpovídají sloupečkům v danné tabulce. Laravel automaticky přiřazuje hodnoty z tabulek odpovídajícím proměnným v modelu, díky čemuž můžeme jednoduše upravovat data v databázi.

Příkladem může být model UserBalance, který musí pracovat s dvěma proměnnými z tabulky user_balance: userId a balance. Tabulka user_balance má více sloupců než jen userId a balance, ale ostatní informace není třeba upravovat nebo s nimi jakkoliv pracovat, tudíž není potřeba je mít v modelu, jelikož se vyplní automaticky.

```
1 class UserBalance extends Model
2 {
3     use HasFactory;
4
5     protected $fillable = [
6         'user_id', // Add user_id to the fillable array
7         'balance',
8     ];
9 }
```

Listing 22: Models/UserBalance.php

Modely jsou umístěny ve složce app/Models a mohou být vytvořeny pomocí příkazu 'php artisan make:model názevModelu'. Mějme například model BuyCryptoCurrency. Poté, co si pomocí php nastavíme, jak bude tabulka v databázi vypadat, uděláme tzv. migraci. Pomocí příkazu php artisan migrate se v databázi vytvoří tabulka podle daného modelu. V našem případě se tabulka automaticky pojmenuje buy_crypto_currencies a díky tomu, že náš model extenduje třídu model předvytvořenou od laravelu lehce se pak pomocí modelu s databází interaguje.

Díky modelům lze jednoduše provádět dotazy na databázi pomocí metod jako je například 'create()'. Jedna z použitých funkci, jež provadí dotazy na databázi, je addToMyList. Tato funkce zajišťuje přidání kryptoměny do korespondující tabulky favouriteCryptoCurrencies. Pomocí Auth::id() se získá ID aktualně přihlášeného uživatele. Následně pomocí metody request získáme jméno kryptoměny, kterou si uživatel přidal do oblíbených.

```
function addToMyList(Request $request) {
        $userId = Auth::id();
        $cryptocurrencyName = strtolower($request->input('cryptocurrencyName'))
    ;

        if($this->doesRecordExist($cryptocurrencyName, $userId) === false) {
            FavouriteCryptoCurrency::create([
                'user_id' => $userId,
                'name' => $cryptocurrencyName,
            ]);
        }
    }
```

Listing 23: Controllers/AddToMyListController.php

6.2.2 Schéma databáze

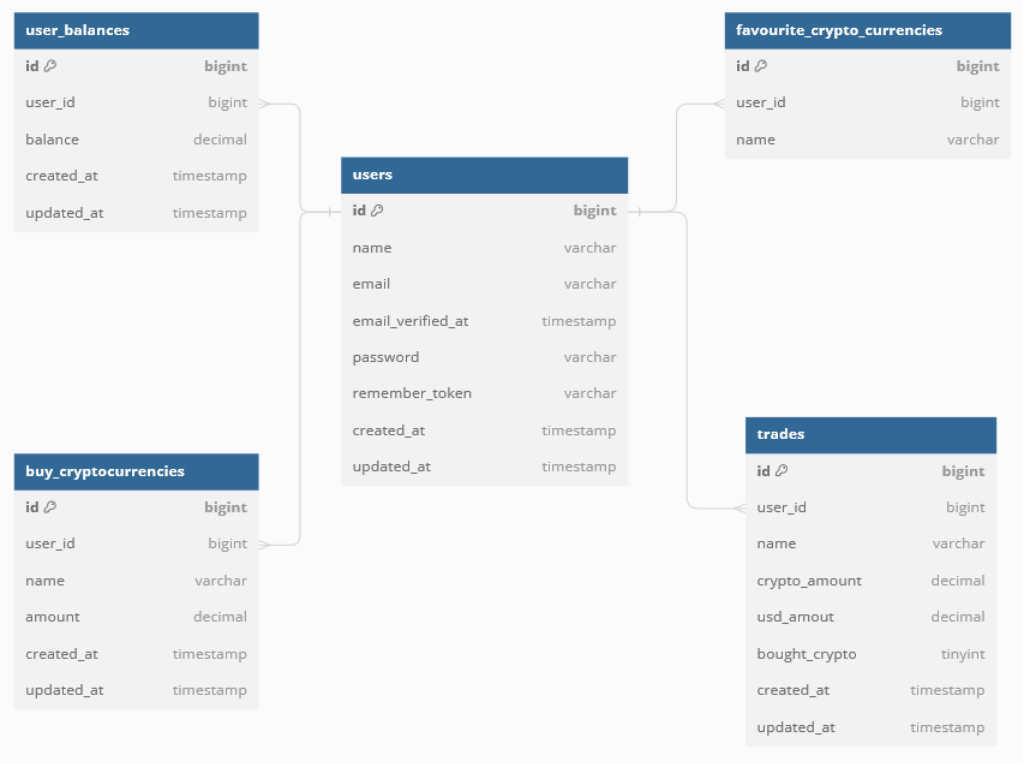
Naše databáze je složena z 5 tabulek a to jsou: users, user balances, buy cryptocurrencies, favourite crypto currencies a trades. Každá tabulka obsahuje primární klíč ID, který se u hlavní tabulky users váže na všechny ostatní tabulky pomocí sloupců s cizím klíčem user id.

Hlavní tabulka users obsahuje uživatelské jméno, email, heslo uživatele a tzv. remember token, který říká zda-li chce uživatel zůstat přihlašení pro budoucí použití aplikace.

Tabulka trades slouží jako celkový seznam všech obchodů, které uživatel provedl. Částky, jména kryptoměn, počet koupených tokenů a další důležitá data.

Tabulka buy_cryptocurrencies následně slouží k zaznamenání celkové aktuální částky dané kryptoměny, se kterou uživatel provedl nějakou transakci. Tabulka user_balances slouží k zaznamenávání aktuálních stavů účtů uživatelů v dolarech a poslední tabulka favourite_crypto_currencies slouží k ukládání kryptoměn, jež si uživatel označí jako oblíbené.

Všechny výše zmíněné tabulky, až na favourite_crypto_currencies obsahují také časový údaj. U této tabulky jsme usoudili, že není třeba udávat čas, kdy si uživatel přidal kryptoměnu do oblíbených.



6.3 Controllery

V controllerech se odehrává hlavní logika naší webové aplikace. V této kapitole je stručně zmíněno, za co je jaký controller zodpovědný. První controller AddMoneyController slouží k přidávání finančních prostředků uživatelem.

```
class AddMoneyController extends Controller {
    function addMoney(Request $request) {
        $this->validateData($request);
        $userId = Auth::id();
        $usdAmount = $request->input('amount');
        $userBalance = UserBalance::where('user_id', $userId)->first();
        $newUserBalance = $userBalance->balance + $usdAmount;
        $userBalance->balance = $newUserBalance;
        $userBalance->save();

    }
    function validateData(Request $request) {
        $request->validate([
            'amount' => ['required', 'numeric', 'gt:0'],
        ]);
    }
}
```

Listing 24: Controllers/AddToMyListController.php

Na tomto příkladu můžeme vidět, že pomocí ORM získáme stav konta uživatele, poté k němu přičteme počet peněz, které uživatel vyplnil a následně tento záznam uložíme.

AddToMyListController se stará o přidávání a odebírání kryptoměn z oblíbených a pomocí modelu FavouriteCryptoCurrency interaguje s tabulkou v databázi.

BuyCryptoCurrencyController slouží jak ke koupi tak k prodeji kryptoměn. Mění kolik má uživatel peněz a kryptoměn a také ukládá trade, aby si uživatel mohl zobrazit celou historii tradů.

CryptoDetailController zodpovídá za získávání potřebných dat ke kryptoměně, kterou si uživatel vyhledal. Získává a formátuje tedy data u krypto burz.

DashboardController získává data z krypto burzi ke všem kryptoměnám, které má uživatel v oblíbených. Extenduje CryptoDetailController a využívá některé jeho metody. IndexController slouží převážně k získání dat pěti kryptoměn, které se uživateli zobrazí v tabulce na uvítací stránce.

ProfileController byl předem vytvořený frameworkem Laravel a stará se o to, že mění údaje v uživatelském účtě. Uživatel si může změnit email, jméno a heslo. Může si také učet smazat a založit nový.

UserBalanceController Získává pomocí modelů z databáze všechny kryptoměny, které uživatel vlastní, oblíbené kryptoměny uživatele, stav konta uživatele a historii tradů uživatele.

7 Závěr

Cílem bylo vytvořit webovou aplikaci, která by umožnila uživateli si procvičit obchodování na burze s kryptoměnami pomocí nerealných peněz. Tento cíl se dle našeho názoru podařilo splnit. Aplikace nabízí obchodování s mnohými kryptoměnami, grafy pro vývoj cen těchto kryptoměn, možnost personalizace stránky uživatelem pomocí přidávání oblíbených kryptoměn a způsob kterým může uživatel sledovat úspěšnost svého investování pomocí spočteného portfolia v záložce balance.

Seznam použitých zdrojů

"Laravel - Script @php artisan package:discover handling the post-autoload-dump event returned with error code 255"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/46986001

"Neos Documentation - Docker and Docker Compose Setup"[online]. Neos Documentation, [cit. 2024-04-25]. Dostupné z:
https://docs.neos.io/guide/installation-development-setup

"Laravel Documentation - Laravel Sail"[online]. Laravel, [cit. 2024-04-25]. Dostupné z:https://laravel.com/docs/10.x/sail

"Unsupported operating system with Docker on Windows 10 with WSL2 - Stack Overflow"[online]. Stack Overflow, 2021 [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/66412753

"Laravel 6 Tutorial for Beginners #15 - Eloquent Models - YouTube"[online]. YouTube, [cit. 2024-04-25]. Dostupné z:
https://www.youtube.com/watch?v=1lyu1KKwC74

"Laravel Tutorial #20 - Models Explained"[online]. YouTube, [cit. 2024-04-25]. Dostupné z: https://www.youtube.com/watch?v=hCZ2bu_0khg

"Tailwind in 100 Seconds"[online]. YouTube, [cit. 2024-04-25]. Dostupné z: https://www.youtube.com/watch?v=mr15Xzb1Ook

"Tailwind CSS Tutorial #12 - Grids"[online]. YouTube, [cit. 2024-04-25]. Dostupné z: https://www.youtube.com/watch?v=_r2qB44o_Fs

"The BEST Way to Create Responsive Design with Tailwind CSS (2023)"[online]. YouTube, [cit. 2024-04-25]. Dostupné z: https://www.youtube.com/watch?v=PuovsjZN11Y

"Vue 'export default' vs 'new Vue'"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/48727863

"You're doing dark mode wrong!"[online]. YouTube, [cit. 2024-04-25]. Dostupné z: https://www.youtube.com/watch?v=WTchW0LdWL0

"Change body background with data-theme attribute on click with React"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/72740277

"Typewriter Animation in CSS"[online]. YouTube, [cit. 2024-04-25]. Dostupné z: https://www.youtube.com/watch?v=yefgBA1CecI

"Laravel Dynamic Routes: A Comprehensive Guide"[online]. Medium, [cit. 2024-04-25]. Dostupné z: https://medium.com/@sifztech/4d8b9f53af10

"Cannot connect to Laravel MySQL database"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/39572832

"How To Use Inertia.js in Your Laravel Projects"[online]. Kinsta Blog, [cit. 2024-04-25]. Dostupné z: https://kinsta.com/blog/laravel-inertia/

"Tailwind CSS Documentation"[online]. [cit. 2024-04-25]. Dostupné z: https://tailwindcss.com/docs/

"PHP Artisan Migrate throwing PDO Exception - could not find driver using Laravel"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/22463614

"PHP Laravel No connection could be made because the target machine actively refused it"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/32514241/

"Laravel Docker - The stream or file "/var/www/html/storage/logs/laravel.log"could not be opened: failed to open stream: Permission denied"[online]. Stack Overflow, [cit. 2024-04-25]. Dostupné z: https://stackoverflow.com/questions/50552970/

"MySQL :: MySQL 8.0 Reference Manual :: 11.3 Date and Time Types"[online]. MySQL, [cit. 2024-04-25]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/datetime.html

"TW Elements - Hiding Elements"[online]. TW Elements, [cit. 2024-04-25]. Dostupné z: https://tw-elements.com/learn/te-foundations/te-ui-kit/hiding-elements/

Listings