Gymnázium, Praha 6, Arabská 14

Programování

Ročníková práce

Duben 2025 Gabriela Stamenova

Gymnázium, Praha 6, Arabská 14

Arabská 14, Praha 6, 160 00

Ročníková práce

Předmět: Programování

Téma: Budget Tracker

Školní rok: 2024/2025

Autoři: Gabriela Stamenova

Třída: 4.E

Vedoucí práce: Mgr. Jan Lána

Třídní učitel: Mgr. Blanka Hniličková

Čestné prohlášení

Prohlašuji, že jsem jediným autorem 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 uděluji bezúplatně škole Gymnázium, Praha 6, Arabská14 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..... dne.....

Podpis:

Anotace

Cílem mé práce je vytvořit uživatelsky přívětivou webovou aplikaci pro správu osobních financí. Aplikace Budget Tracker umožní uživatelům sledovat své příjmy a výdaje, analyzovat finanční toky a zlepšovat správu osobního rozpočtu. Některé hlavní funkce aplikace jsou: zaznamenávání příjmů a výdajů, kategorzace transakcí, vizualizace dat ve formě grafu a další. Program bude postaven na platformě SvelteKit, což je tzv. Framework, který kombinuje více programovacích jazyků jako například JavaScript, HTML a CSS.

Abstract

The aim of this work is to create a user-friendly web application that would serve as a clear and efficient environment for organizing personal finances. Application Budget Tracker allows the user to follow their incomes and outcomes, analyze financial flow and improve the management of one's personal budget. Some main functions of the application are: keeping record of incomes and expenses, categorization of transactions, visualization of data with the help of a graph etc. The program will be built on a platform called Sveltekit, which is so called Framework that combines multiple programming languages such as JavaScript, HTML and CSS.

Obsah

1 Úvod

Tato webová aplikace má za úkol sloužit jako pomocník při sledování osobních financí. Nabízí uživatelům různé funkce jako zaznamenání příjmů a výdajů, přidání popisu k transakcím pro lepší přehled, analýza zaznamenaných dat pro předpověď budoucích výdajů a tak dále. Po spuštění aplikace si musí uživatel vytvořit účet a následně bude vyžadována autorizace e-mailu. Poté, co má uživatel již vytvořený účet, musí do něj přihlašovat a jeho data budou vždy uložena. Pro vybudování aplikace jsem použila Framework Sveltekit a databázi PocketBase.

1.1 Zadání

Zadání mé ročníkové práce bylo:
Vytvořit webovou aplikaci, která umožňuje sledování osobních financí. Uživatel se přihlásí (vytvoří si účet) a přidá svůj příjem - bude možnost přidat zdroj a popisek ke každému příjmu. Program bude obdobně manipulovat i s výdaji - uživatel průběžně zadává svoje aktivní výdaje s možností přidání popisku. Celkový rozpočet se bude regulovat v závislosti na aktivních příjmech a výdajích. Pokud zbyde čas, tak se pokusím udělat i graf, který bude reprezentovat osobní finance uživatele za určité období (měsíc). Program bude také nabízet budoucí příjmy/výdaje na základě dat přidaných uživatelem.

2 Použité technologie

K vyhotovení aplikace jsem použila intergrované vývojové prostředí Visual Studio Code, což je editor zdrojového kódu vyvíjený společností Microsoft pro operační systémy Windows, Linux a macOS. [1] Pro programování samotné aplikace jsem zvolila vývojovou platformu Sveltekit kvůli její praktičnosti a efektivitě. Sveltekit je poměrně neobvyklý JavaScriptový framework, ale vzhledem k tomu, že pro letošní ročníkovou práci jsem chtěla použít technologii, kterou jsem dříve neznala, tak jsem si prohlédla různé platformy a Sveltekit mi přišel jako nejlepší a nejefektivnější volba, protože nemá runtime a je velmi rychlý, umožňuje generování statických aplikací při buildu a má zabudovaný tzv. routing, což je systém routování založený na souborové struktuře - URL adresy odpovídají struktuře složek v projektu. Sveltekit používá několik programovacích jazyků např. JavaScript, HTML, CSS a frontend-ový framework Svelte. [2] Pro snadnou manipulaci s daty z databáze jsem použila DB PocketBase. Byla mi doporučená z hlediska jednoduchosti, rychlého nasazení a tzv. WebSocketů pro sledování změn v databázi v reálném čase. K vypracování dokumentace jsem použila LaTeX Overleaf.

Obr. 1 - Loga použitých nástrojů

3 Struktura programu

Program je rozdělen do tří hlavních sekcí: *Přehled, Příjmy a Výdaje*. Domovská stránka *Přehled* poskytuje shrnutí aktuálních příjmů a výdajů, které si uživatel zadal, aktuální disponibilní zůstatek, a také vytváří horizontální sloupcový graf, který udává top 10 nejvyšších příjmů/výdajů za určité období. Snažila jsem se udělat i funkci pro předpověď budoucích transakcí, která se v rámci možností. Z hlediska kódu je program rozdělen na několik částí. V následující kapitole bych ráda popsala obsah složky `src`, kde se nachází naprostá většina logiky kódu.

3.1 Routes

Takzvané routy jsou v programování využívány k definování jednotlivých stránek a API endpointů v aplikaci. API endpointy jsou specifické URL adresy, které slouží k výměně dat mezi frontendem a backendem. SvelteKit používá tzv. `file-based routing`, což znamená, že struktura souborů ve složce `src/routes` automaticky určuje dostupné stránky a API trasy. Nyní se podíváme na sloužku `Routes`.

3.1.1 Expenses

Ve složce `expenses` se nachází dva soubory: `+ page.server.js` a `+ page.svelte`. (Důležitá poznámka: soubory začínající + jsou uživatelem vytvořené API endpointy). Třída `+ page.server.js` zodpovídá za vkládání a aktualizaci výdajů. Na začátku je funkce *load()*, která má za úkol načíst data při otevření stránky. Dále kontroluje, zda je uživatel přihlášen, a pokud ne, tak ho přesměruje na přihlašovací stránku `Login`. Poté načte seznam výdajů z databáze, setříděný sestupně podle atributů *date, source, amount*. Funkce *insertRecord()* umožňuje přidání nového výdajového záznamu. Dále převádí formulářová data na objekt a zpracovává checkbox *isReccuring* (opakující se platby) a ukládá výdaj do databáze. Po uložení načte záznam o zůstatku uživatele a odečte uložený výdaj, poté aktualizuje kapitál uživatele. Další důležitou funkcí je *updateRecord()* - aktualizace již existujícího výdaje.

const recordId = url.searchParams.get('recordId');

const oldAmount = Number(url.searchParams.get('oldAmount'));

const difference = Math.abs(data.amount - oldAmount);

const isIncreasing = data.amount > oldAmount;

Tento kus kódu ukazuje, jak program postupuje při aktualizaci výdaje. Nejprve získá ID záznamu a původní částku, poté spočítá rozdíl mezi novou a starou částkou a zjistí, zda se zůstatek zmenšuje nebo zvětšuje, a upraví záznam v `expenses`. Aktualizuje zůstatek buď odečtením nebo přičtením částky (toho rozdílu staré a nové částky).

Druhá třída ve složce `expenses` je `+ page.svelte`, která má za úkol zobrazovat a spravovat výdaje uživatele, a zároveň umožňuje jejich filtrování, vyhledávání a přidávání nových záznamů. Hlavní funkce jsou:

Načítání dat: Přijímá data (records – výdaje, categories – kategorie) přes export let data. Data se ukládají do `recordsStore` pro sdílení v aplikaci.

Vkládání nového výdaje: Formulář v levé části umožňuje zadat: datum (date), zdroj (source), částku (amount), kategorii (category), popis (description), označit jako opakující se platbu (isReccuring).

Filtrování a vyhledávání: Vyhledávání se provádí podle zdroje (source) a popisu (description). Filtrování podle kategorie: kliknutím na tlačítko. Filtr částky: od-do. Debounce: pro plynulé filtrování bez zbytečných requestů (setTimeout 500ms). Automaticky nastavuje maximální hodnotu pro výběr data (dateInput).

Reset filtrů: Tlačítko "Resetovat" vynuluje všechny filtry.

3.1.2 Incomes

Soubor `Incomes` má podobné funkce jako `Expenses`, akorát pracuje s příjmy místo s výdajů. Nejprve probíhá autorizace uživatele, pokud není definován, provede se přesměrování na Login. Po přihlášení se načte záznam příjmů, který je seřazen podle data (date) - nejnovější první, zdroje (source) - abecedně, částky (amount) - největší částka první. Dále tato třída umožňuje vložení nového příjmu. Uživatel zadá data do formuláře, který je následně převeden na objekt a vložen do databáze. Poté se aktualizuje zůsatek tím, že se přičte příslušná částka.

Dále lze také upravit již existující příjmy. Princip je stejný jako u výdajů - program porovná novou částku se starou a spočítá rozdíl. Následně změnu zaznamená v `Incomes` a aktualizuje zůstatek.

3.1.3 Login

Nesdílnou součástí programu je přihlášení. Kód, který se o něj stará se nachází ve třech třídách v rámci routy `Login`.

První je `+error.svelte`. Ta zodpovídá za zachycování chyb, které mohou nastat během přihlášení.

Druhá třída se jmenuje `+page.server.js` a má za úkol zajistit autentizaci uživatele pomocí databáze PocketBase. Nejprve kontroluje, zda je uživatel přihlášen. Pokud ano, provede redirect na hlavní stránku. Pokud ne, stránka se normálně načte a vyžaduje přihlášení - načte formulářová data, a poté je převede na objekt. Následně probíhá autentizace v databázi, kdy program kontroluje, zda e-mail a heslo jsou správné. Pokud ano, uživatel je přesměrován na hlavní stránku, pokud ne, vrací chybu.

Třetí třída obsažena v tomto souboru je `+page.svelte`. Ta zobrazuje přihlašovací formulář s poli Email a Heslo (viz. Obr. 2). Obsahuje také odkaz na registraci pro uživatelé bez účtu. Po registraci je uživatel vytvořen v databázi a může se přihlásit. Stylování je uskutečněno pomocí Tailwind CSS.

Obr. 2 - Ukázka Loginu

3.1.4 Register

V případě, že uživatel nemá vytvořený účet, tak ho program nasměruje k registračnímu formuláři. Ten vyžaduje email, heslo a jméno, aby následně uložil uživatele do databáze. Poté se může normálně přihlásit přes Login.

3.1.5 +page.server.js

Třída `+page.server.js` zodpovídá za načtení dat pro dashboard uživatele neboli titulní stránku aplikace. To zahrnuje údaje o aktuálním zůstatku, posledních 10 příjmů a výdajů, predikce budoucích příjmů a výdajů na základě uložených dat a horizontální sloupcový graf, který udává finanční aktivitu za určitou dobu.

Funguje to tak, že program zkontroluje, zda je uživatel přihlášen, následně načte data z databáze, zpracuje příjmy a výdaje (součty, predikce,...), a poté vrátí upravená data pro zobrazení na frontendu.

```
const res_chartExpenses = await locals.pb.collection('expenses').getFullList({
        filter: `user = "${userId}" && date >= "${twoMonthsAgo}"`,
        sort: '-date,-amount',
        fields: 'amount,date,source',
        requestKey: null
});

const groupedChartExpenses = res_chartExpenses.reduce((acc, expense) => {
        const source = (expense.source || 'Neznámý zdroj').trim().replace(/\s+/g, ' ').toLowerCase();

        if (!acc[source]) {
                acc[source] = { source: expense.source.trim().replace(/\s+/g, ' '), amount: 0 };
        }

        acc[source].amount += expense.amount;
        return acc;
}, {});
```

Tento kus kódu ukazuje, jak probíhá agregace dat pro graf. Nejdříve probíhá skupinování příjmů podle zdroje, poté počítání celkové částky pro každý zdroj, a pak program seřadí údaje podle výše příjmů/výdajů.

```
const detectStablePaycheck = (incomes) => {
        const frequency = {};
        incomes.forEach(({ amount }) => {
                frequency[amount] = (frequency[amount] || 0) + 1;
        });
        const stablePaychecks = Object.keys(frequency)
                .map(Number)
                .filter((val) => frequency[val] > 1);
        return stablePaychecks.length > 0 ? Math.max(...stablePaychecks) : 0;
};

const getWeightedAverage = (incomes) => {
        if (incomes.length === 0) return 0;
        incomes.sort((a, b) => new Date(b.date) - new Date(a.date) || b.amount - a.amount);
        const last5 = incomes.slice(0, 5);
        let weights = last5.map((_, index) => last5.length - index);
        let weightedSum = last5.reduce((sum, t, index) => sum + t.amount * weights[index], 0);
        let totalWeight = weights.reduce((sum, w) => sum + w, 0);
        return weightedSum / totalWeight;
};
```

Tenhle kus kódu ukazuje, jak funguje predikce příjmů a výdajů. Nejprve program zjistí, zda má uživatel pravidelný příjem (výplata, kapesné,...), následně spočítá vážený průměr pro proměnlivé příjmy a výdaje

```
const stablePaycheck = detectStablePaycheck(res_pastIncomes);
const variableIncomes = res_pastIncomes.filter((t) => t.amount !== stablePaycheck);
const predictedVariableIncome = getWeightedAverage(variableIncomes);
const predictedIncome = stablePaycheck + predictedVariableIncome;

const stableRecurringExpense = detectStablePaycheck(res_pastExpenses);
const variableExpenses = res_pastExpenses.filter((t) => t.amount !== stableRecurringExpense);
const predictedVariableExpense = getWeightedAverage(variableExpenses);
const predictedExpense = stableRecurringExpense + predictedVariableExpense;

const insufficientData = res_pastExpenses.length < 5 && res_pastIncomes.length < 5;
```

Dále předvídá budoucí finanční aktivity na základě historických dat. Používá k tomu dvě hlavní metody: detekce pravidelných částek (např. výplata, pravidelný nájem, předplatné, atd.) a provádí vážený průměr proměnlivých částek - počítá, jak se mění nepravidelné příjmy a výdaje v čase a na základě toho hádá budoucí hodnoty. Jak to funguje? Program seřadí transakce podle data, následně každé transakci přiřadí váhu (čím novější, tím vyšší váha), a poté spočítá vážený průměr. Nakonec se predikované hodnoty sečtou. Tahle funkce je dobrá k tomu, že pomáhá uživateli vidět, jaké příjmy a výdaje může očekávát, případně upozorní na možné finanční problémy (pokud očekáváné výdaje jsou větší než očekávané příjmy), a celkově snadňuje finanční plánování a rozpočet.

3.2 App.css

V této třídě se nachází CSS stylesheet, který definuje globální styly pro webovou aplikaci. Obsahuje různé vizuální prvky (např. barvy, fonty, rozvržení, pozadí atd.). Dále zodpovídá za to, že se pozadí nehýbe při scrollování a adaptuje se podle daného rozvržení obrazovky (responzivita). Zajímavou součástí této třídy, se kterou jsem doposud nepracovala, je *.visually-hidden*, což se používá pro skrytí textu na obrazovce. Uživatel ho nevidí, ale je čitelný pro případné asistivní technologie.

3.3 Lib

Další poměrně důležitou částí mého programu je soubor *lib*, který obsahuje podsoubory, které zodpovídají za různé nezávislé komponenty programu. V této kapitole popíšu nejpodstatnější z nich.

3.3.1 Components

Zde se nachází několik dalších podsouborů, které plní různé funkce.

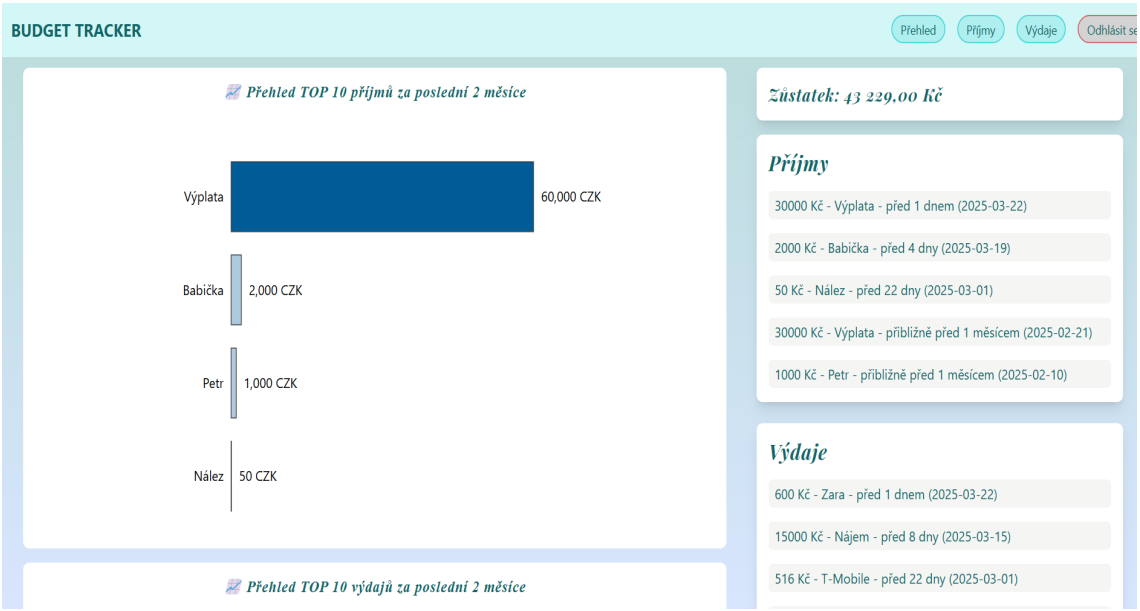
Dashboard vytváří `D3.js` sloupcové grafy příjmů a výdajů na titulní straně a před tím kontroluje, zda je dostatek dat pro jejich vyhotovení. V případě nedostatku dat se zobrazí šedý box s upozorněním.

Třída *Ai.Detail.svelte* umožňuje zobrazení, úpravu či mazání transakcí. Při mazání zobrazuje dodatečnou zprávu, zda chce uživatel doopravdy transakci smazat. Zobrazí datum transakce v českém formátu (dd.mm.yyyy). Pokud chybí popis, zobrazí "Žádný popis transakce". Tlačítko „Upravit“ otevře formulář v modálním okně. Tlačítko „Smazat“ zobrazí potvrzovací dialog.

UpdateRecordModalForm.svelte je další třídou v souboru *Components* a slouží k úpravě existujícího záznamu transakce v modálním okně. 1Program zobrazuje modální okno pro editaci transakce, předvyplní formulář aktuálními údaji o transakci, umožňuje uživateli změnit datum, částku, zdroj, kategorii a další detaily, pak odesílá formulář a po úspěšné aktualizaci se data uloží a okno se zavře.

4 Databáze

Jak jsem již zmínila, použila jsem databázi PocketBase. Ta se využívá při ukládání, úpravě a mazání finančních transakcí a dále při ukládání zaregistrovaných uživatelů. Jak to funguje? Data se ukládají do `recordsStore` (napojení Svelte store a PocketBase) a dále se zobrazují v grafech a tabulkách. Když uživatel přidá transakci, program odesílá POST požadavek na PocektBase API. Poté PB automaticky vygeneruje ID a uloží transakci, následně probíhá aktualizace `RecordsStore`, aby se změny okamžitě projevily v UI. PocketBase jsem zvolila z důvodu jeho efektivity a absence složitého serverového nastavení.



Obr. 3 - Titulní stránka programu

5 Komplikace

Při vytváření toho projektu jsem narazila na mnoho problémů.

Na začátku jsem si nevěděla rady, jak vyřešit autentizaci a správu uživatelů pomocí `cookies`. Nakonec jsem zhlédla různá videa na YouTube, abych si vyjasnila, jak to funguje. [3]

Dále jsem se poměrně dlouho zabývala reaktivitou Svelte. Cílem bylo, aby se data aktualizovala při každém načtení formuláře. Ze začátku proces trval dost dlouho, ale po optimalizování aplikace tento problém zmizel.

Dalším závažným problémem, na který jsem narazila byla práce s D3.js vizualizací. To byla pro mě nová technologie, takže jsem opět strávoila nějaký čas tím, že jsem zjišťovala, jak to funguje. [4] Poté mi to moc nešlo implementovat (konkrétně zprovoznit grafy), ale poté, co jsem si k tomu sedla o týden později, tak se to podařilo.

Dále jsem strávila poměrně dost času nad vymyšlením matematického postupu, který predikuje budoucí finanční aktivity. První myšlenka byla aritmetický průměr, což je nebyl velice racionální postup. Pak jsem přišla na vážený průměr, kde hraje roli to, zda je platba opakující se, a pokud ano, tak má větší váhu než ty jednorázové.

6 Závěr

Výsledkem mé práce je webová aplikace, která slouží uživateli k přehledu jeho osobních financí. Je určena zejména (ale nejen) dětem a studentům, kteří často mají užší finanční kapitál a nechovají se k němu zodpovědně. Často se mohou ocitnout v situaci, že nevědí, kam jim peníze zmizely. Tato webová aplikace poskytuje primitivní nástroj pro sledování finanční aktivity a lepší plánování útrat. Práce splnila očekávání a myslím si, že je vzhledem k časovým možnostem a ostatním povinnostem povedená. Nejspíše se jedná o moji poslední práci tohoto typu, alespoň v následujících letech, protože prozatím mám jiné plány pro svoji budoucnost a to v oblasti vědy. I přes to jsem ráda za všechny nové znalosti, které jsem pomocí této práce nasbírala, a za všechny oblasti, ve kterých mě programování zlepšilo (časová efektivita, následky rozhodnutí, vytrvalost, atd.). Každopádně práci považuji za úspěšnou a zadání za splněné.

7 Seznam obrázků

8 Zdroje

8.1 Informační zdroje

1. Visual Studio Code [Online] [Citace: 6. 03. 2025] https://cs.wikipedia.org/wiki/Visual_Studio_Code

2. Sveltekit [Online] [Citace: 6.03.2025] https://svelte.dev/

3. SvelteKit Authentication Using Cookies [Citace: 15.03.2025] https://www.youtube.com/watch?v=E3VG-dLCRUk

4. Začínáme s D3.js [Online] [Citace: 15.03.2025] https://www.tempmail.us.com/cs/d3js/nastaveni-pracovniho-prostredi-d3-js-pomoci-html-javascriptu-a-node-js#google_vig

8.2 Zdroje obrázků

1. Visual Studio Code: https://www.google.com/search?q=visual+studio+code+logo&sca_esv=1df9984c226b3a13&udm=2&biw=1536&bih=695&ei=9FMnZtGLA5r-i-gPzvytmAI&ved=0ahUKEwjR1ZrG2deFAxUa_wIHHU5-CyMQ4dUDCBA&uact=5&oq=visual+studio+code+logo&gs_lp=Egxnd3Mtd2l6LXNlcnAiF3Zpc3VhbCBzdHVkaW8gY29kZSBsb2dvMgUQABiABDIEEAAYHjIEEAAYHjIEEAAYHjIEEAAYHjIEEAAYHjIEEAAYHjIEEAAYHjIEEAAYHjIGEAAYBRgeSKUcUJsHWN0acAJ4AJABAJgBbqABuQSqAQM0LjK4AQPIAQD4AQGYAgegAqQEwgIKEAAYgAQYQxiKBcICBhAAGAcYHpgDAIgGAZIHAzQuM6AHgxw&sclient=gws-wiz-serp&safe=active&ssui=on#vhid=PryLAM8_xQK-gM&vssid=mosaic

2. Latex Overleaf: https://www.google.com/url?sa=i&url=https%3A%2F%2Fde.wikipedia.org%2Fwiki%2FOverleaf&psig=AOvVaw0Klf6nVSrPerAjA9nJHaAn&ust=1713939993773000&source=images&cd=vfe&opi=89978449&ved=0CBAQjRxqFwoTCNDs65fa14UDFQAAAAAdAAAAABAZ

3. SvelteKit: https://www.google.com/url?sa=i&url=https%3A%2F%2Fgithub.com%2Fnavneetsharmaui%2Fsveltekit-starter&psig=AOvVaw2pXo-7F5n8pQpcgQ13KAVs&ust

=1742803692189000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCPDsv-nf
n4wDFQAAAAAdAAAAABAJ

5. Javascript: https://www.google.com/url?sa=i&url=https%3A%2F%2Fcommons.wiki
media.org%2Fwiki%2FFile%3AJavascript_badge.svg&psig=AOvVaw1FP0bCZb
evTyQS07_WoUru&ust=1713940210534000&source=images&cd=vfe&opi=89978449&
ved=0CBAQjRxqFwoTCOCVw4rb14UDFQAAAAAdAAAAABAE

6. PocketBase https://www.google.com/url?sa=i&url=https%3A%2F%2Fapps.umbrel.com
%2Fapp%2Fpocketbase&psig=AOvVaw1y0ub6LIsaKK6-_-kdI9Fg&ust=1742803881280000&
source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCLCCjcTgn4wDFQAAAAAdAAAAABAE