Opis projektu

Plan projektu: Aplikacja web do zarządzania zbiórkami klasowymi

Założenia organizacyjne

- Projekt oparty na **Next.js**, **PocketBase** i **shadcn, NodeJs, ntfy.sh**.
- wykupiony serwer VPS dla każdej z grup
- projekt dostępny pod adresem domenowym (ja konfiguruję domenę) - jeśli grupa 5 osobowa
- Każda grupa (4-5 osób) dzieli zadania między członków, a lider zespołu (Project Manager) zajmuje się koordynacją i wsparciem technicznym.
- jeśli kilka osób pracuje nad jedną aplikacją tworzymy feature branch i na koniec prac scalamy projekt rozwiązując ewentualne konflikty

1. Tworzenie i zarządzanie zbiórkami

- **Opis funkcji:** Skarbnik tworzyć nowe zbiórki, ustalając ich cel (np. wycieczka, zakup prezentu, impreza klasowa), kwotę do zebrania, termin zakończenia oraz opcjonalne szczegóły (np. za co odpowiada zbiórka). Skarbnik pełni rolę organizatora i administratora zbiórek.
- **Przykład użycia:** Skarbnik klasy tworzy zbiórkę na wycieczkę klasową z celem 1500 zł i ustawia deadline wpłat na dwa tygodnie przed datą wyjazdu.

2. Lista uczestników zbiórki

- **Opis funkcji:** Skarbnik może dodawać uczestników (np. uczniów) i przypisywać im kwotę wpłaty.
- **Przykład użycia:** Skarbnik dodaje do zbiórki listę uczniów i ustawia sugerowaną wpłatę na 50 zł/osobę.

3. Rejestrowanie wpłat i potwierdzenia

- **Opis funkcji:** Skarbnik rejestruje wpłaty uczestników w systemie. Można zaznaczyć wpłaty gotówkowe oraz dodać notatki o osobach, które jeszcze nie wpłaciły.
- **Przykład użycia:** Uczeń wpłaca 50 zł gotówką na zebranie, a skarbnik aktualizuje dane w aplikacji, potwierdzając wpłatę.

4. Powiadomienia o statusie zbiórki

- **Opis funkcji:** Automatyczne powiadomienia dla uczestników o statusie zbiórki, zbliżającym się terminie wpłaty czy zakończeniu zbiórki.
- **Przykład użycia:** **ntfy.sh** oraz bot Discord przypominający codziennie o 19:00 osobom, które nie zapłaciły i o nowych zbiórkach (w momencie utworzenia) oraz możliwość ręcznego wysłania powiadomienia przez skarbnika

5. Historia i raporty

- **Opis funkcji:** Dostęp do historii zakończonych zbiórek z możliwością wygenerowania raportu z podsumowaniem wpłat oraz z ewentualną sumą, która została w kasie.
- **Przykład użycia:** Skarbnik generuje raport z podsumowaniem zbiórki na wycieczkę, aby przedstawić go rodzicom podczas zebrania.

6. Interaktywna tablica postępów

- **Opis funkcji:** Wyświetlanie graficznego postępu zbiórki (np. pasek postępu, koło wypełnienia) oraz kwoty brakującej do realizacji celu.
- **Przykład użycia:** Uczestnicy widzą, ile osób musi jeszcze wpłacić na zbiórkę, ale osoby pozostają anonimowe.

8. Komentarze i sugestie

- **Opis funkcji:** Uczestnicy zbiórki mogą dodawać komentarze i sugestie dotyczące przeznaczenia funduszy.
- **Przykład użycia:** Uczeń proponuje, aby zrezygnować z atrakcji na wycieczce, aby obniżyć jej koszt

9. Moduł zgłoszeń problemów

- **Opis funkcji:** Użytkownicy mogą zgłaszać błędne wpłaty, brakujące informacje lub inne problemy związane ze zbiórką.
- **Przykład użycia:** Uczeń zgłasza, że jego wpłata nie została jeszcze odnotowana.

Różne tryby zbiórek (Dodatkowo)

- **Opis funkcji:** Możliwość wyboru trybu zbiórki:
  - **Publiczny:** Każdy widzi szczegóły zbiórki.
  - **Prywatny:** Widoczna tylko dla zaproszonych uczestników.
- **Przykład użycia:** Zbiórka na prezent dla nauczyciela jest ustawiona jako prywatna, widoczna tylko dla uczniów.

Integracja z kalendarzem (Dodatkowo)

- **Opis funkcji:** Automatyczne dodawanie terminów zbiórek do kalendarza google, z przypomnieniami.
- **Przykład użycia:** Uczeń otrzymuje powiadomienie dzień przed terminem końca zbiórki.

Podział kosztów i nadwyżek (Dodatkowo)

- **Opis funkcji:** Jeśli zebrano więcej pieniędzy niż potrzeba, skarbnik może podjąć decyzję, jak rozdysponować nadwyżkę (np. zwrot lub pozostawienie funduszy w kasie klasowej).
- **Przykład użycia:** Zbiórka na wycieczkę kończy się nadwyżką, więc pieniądze są przeznaczone na dodatkowe pamiątki dla uczniów.

Harmonogram projektu

Tydzień 1:

- Przygotowanie **schematu aplikacji**: struktura baz danych, funkcjonalności, widoki UI.
- Tworzenie wstępnej dokumentacji technicznej i opis architektury.
- Konfiguracja serwera na **OVH**.

Tydzień 2:

- Rozpoczęcie prac nad modułami aplikacji zgodnie z podziałem zadań w grupie.
- Pierwsze integracje między frontendem a backendem.

Tydzień 3:

- Dokończenie modułów aplikacji.
- Dodanie powiadomień ntfy.sh oraz bota Discord.

Tydzień 4:

- Finalizacja projektu: dopracowanie funkcji, monitorowanie aplikacji i jej stabilności.

Podział na grupy i role w zespole

1. Lider zespołu (Project Manager)

Odpowiedzialności:

- Koordynacja działań zespołu i rozdzielanie zadań.
- Wstępna konfiguracja i zarządzanie serwerem **OVH**.
- Utrzymywanie dokumentacji technicznej i organizacyjnej poprzez przydzielanie zadań.
- Wsparcie techniczne członków zespołu.
- Utworzenie organizacji na github
- Utworzenie projektu i zarządzanie nim w serwisie do zarządzania projektami (Notion, Trello, Clickup, Monday) i dodanie pozostałych osób

---

2. Backend Developer (PocketBase)

Odpowiedzialności:

- Projekt i konfiguracja bazy danych w **PocketBase**:
  - Kolekcje: `Users`, `Fundraisers`, `Payments`.
  - Relacje między kolekcjami (np. przypisanie użytkownika do zbiórki).

    - Wprowadzenie kolumny skarbnik w zbiórkach (tylko autor może edytować/usunąć).
- Tworzenie endpointów API (REST) do komunikacji z frontendem (JEŚLI BĘDĄ POTRZEBNE)
- Instalacja PocketBase na serwerze.
- Ustawienie usług takich jak ntfy.sh i bot Discord.
- Monitorowanie działania serwera i kontenerów (np. Grafana i Telegraf(influxDB)) → Jeśli w zespole nie ma devops
- Wsparcie integracji z ntfy.sh i botem Discord.

---

3. Frontend Developer (Next.js + shadcn)

Odpowiedzialności:

- Budowa interfejsu aplikacji za pomocą **Next.js** i komponentów **shadcn**:
    - Strona główna z listą zbiórek.
    - Formularz tworzenia nowej zbiórki.
    - Widok szczegółów zbiórki (lista osób, które zapłaciły).
    - Widok profilu użytkownika (np. historia wpłat).
- Obsługa autoryzacji i sesji użytkowników (logowanie/rejestracja).
- Logowanie błędów do InfluxDB
- Integracja z API PocketBase.

---

4. Powiadomienia i automatyzacje (Bot Developer)

Odpowiedzialności:

- **Powiadomienia ntfy.sh:**
    - Konfiguracja systemu do wysyłania powiadomień push o nowych zbiórkach.
    - Integracja z backendem PocketBase.
- **Bot Discord:**
    - Stworzenie bota przypominającego codziennie o 19:00 o niezapłaconych zbiórkach.

- Wysyłanie listy osób, które jeszcze nie wpłaciły, do dedykowanego kanału na Discordzie.
- Obsługa komend (np. sprawdzenie statusu zbiórek).

---

5. Monitorowanie usług (DevOps)

Odpowiedzialności:

- Monitorowanie stanu aplikacji i serwera:
  - Instalacja narzędzi takich jak Grafana i Telegraf(influxDB) do monitorowania zasobów serwera i kontenerów lub podobnych.
  - Konfiguracja logowania błędów serwera i aplikacji (np. logi PocketBase).
- Konfiguracja reverseproxy do obsługi nazw domenowych (https://nginxproxymanager.com/)
- Zarządzanie backupami danych:
  - Regularne tworzenie kopii zapasowych bazy PocketBase.
- Optymalizacja serwera OVH pod kątem wydajności i bezpieczeństwa.

Przykładowa *niekompletna* struktura bazy danych (PocketBase)

Kolekcja: `Users`

Pole	Typ	Opis
`id`	string	ID użytkownika.
`name`	string	Imię i nazwisko użytkownika.
`email`	string	Adres e-mail.

Kolekcja: `Fundraisers`

Pole	Typ	Opis
`id`	string	ID zbiórki.
`title`	string	Tytuł zbiórki.
`description`	text	Opis zbiórki.

`author`	relation	ID użytkownika - autora zbiórki.

Kolekcja: `Payments`

Pole	Typ	Opis
`id`	string	ID wpłaty.
`user`	relation	ID użytkownika wpłacającego.
`fundraiser`	relation	ID powiązanej zbiórki.
`amount`	number	Kwota wpłaty.

Powodzenia! 😊