ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Извештај за проектна задача по предметите “Имплементација на системи со слободен и отворен код” и ”Напреден веб дизајн” на тема:

“Систем за прибирање и менаџирање на одговори од онлајн форм - FormFlow”

Изработила: Моника Тодорова, индекс: 201082

Ментор: д-р Бобан Јоксимоски

Апстракт

Секојдневно во светот на вебот се раѓаат стотици нови веб страници со најразлични цели за маркетинг, поврзување со потенцијални клиенти, продажба и уште многу акции кои се тесно поврзани со комуникација. Со цел да се оствари основна комуникација помеѓу две страни на една веб страна потребна е конфигурација која за лица со недоволно техничко познавање е предизвикувачка за поставување. Поттикната од овој проблем, ја креирав FormFlow апликацијата која овозможува олеснет пристап на конфигурација, а воедно нуди и соодветени опции за прибирање и менаџирање на информациите добиени од онлајн формите кои се поврзани во апликацијата.

Вовед

FormFlow е веб апликација која е дизајнирана да го поедностави процесот на прибирање и менаџирање на одговори од онлајн форми. Им овозможува на корисниците да ги поврзат нивните форми до соодветен ендпоинт, со што сите одговори се прибираат и организираат. Во FormFlow корисниците можат да видат детали за одговорите, да додаваат тагови за полесна категоризација, да филтрираат според таг или статус и да групираат во проекти. Ова ја прави апликацијата соодветна алатка за лица со и без техничко познавање, иако е потребно основно познавање за поврзување на формата. Изработена е користејќи технологии како Laravel за backend, Vue3 за frontend и MySQL база на податоци.

Во продолжение детално ќе разработиме за веб апликации, користените технологии како и начинот на изработка и функционирање на апликацијата FormFlow.

Што е Веб?

Вебот, познат како World Wide Web (WWW) е меѓусебно поврзан систем на јавни веб страници достапни преку Интернет. Вебот е всушност една од многуте апликации изградени врз основа на Интернетот. Системот што ние денес го знаеме како "Веб" се состои од неколку компоненти:

- HTTP протоколот - управува со преносот на податоци помеѓу серверот и клиентот,
- URL - идентификатор кој обезбедува пристап до веб-компонента,
- HTML - најпознат формат за публификување на веб документи,
- Веб сервер - компјутер каде се зачувуваат документи до кои може да се пристапи со користење на HTTP протоколот.

Поврзувањето на различните веб компоненти преку хипер врски е главниот концепт на Вебот. Така всушност е креирана функцијата како збирка од поврзани документи. Вебот е од огромно значење во денешницата. Го поврзува целиот свет и овозможува лесна и непречена комуникација, споделување и пребарување во било кое време.

Веб страница и Веб сајт

Веб страница е документ поставен на Интернет и достапен за разгледување. Веб документите имаат наставка .html или .htm. Содржината на веб страниците е составена комбинирани текстови, слики, звуци и видео записи. За да биде достапна една веб страница, таа мора да има свој уникатен линк за пристап. Линкот, попознат и како URL.

Веб апликација

Веб апликациите користат комбинација од скрипти од страна на серверот за да се справат со складирање и пронаоѓање информации и од страна на клиентот за да се презентираат тие информации. Преку нив се овозможува интерактивност помеѓу понудувачи на услуги и нивните корисници. Апликациите овозможуваат креирање документи, споделување информации, непречена комуникација без оглед на локацијата или уредот.

A. Како работат веб апликациите?
   Еден типичен тек на веб апликација ги содржи следниве чекори:
   1. Корисникот активира барање до веб серверот преку Интернет, поконкретно преку веб прелистувач.
   2. Веб серверот го препраќа барањето до соодветниот сервер за веб апликации.
   3. Веб апликативниот сервер ја извршува бараната задача - како што е пребарување во базата на податоци или обработка на податоци и потоа генерира соодветен повратен одговор.
   4. Серверот за веб апликации ги испраќа резултатите со бараните податоци до веб серверот.
   5. Веб серверот му одговара на клиентот со приказ на бараните информации на екранот.

B. Предности на веб апликациите
   Во последно време веб апликациите заземаат голем замав и влијание во светот на Интернет, а тоа се должи на неколкуте предности со кои тие се истакнуваат:
   1. Веб апликациите работат на повеќе платформи без оглед на оперативниот систем или уредот се додека прелистувачот е компатибилен.
   2. Сите корисници пристапуваат до истата верзија, елиминирајќи ги сите проблеми со компатибилноста.
   3. Тие не се превземаат на уредот, со што се елиминираат ограничувањата на меморија во уредот.
   4. Ги намалуваат трошоците за бизнисот и за крајниот корисник бидејќи се полесни за одржување и не бараат специфична хардверска конфигурација од крајниот корисник.

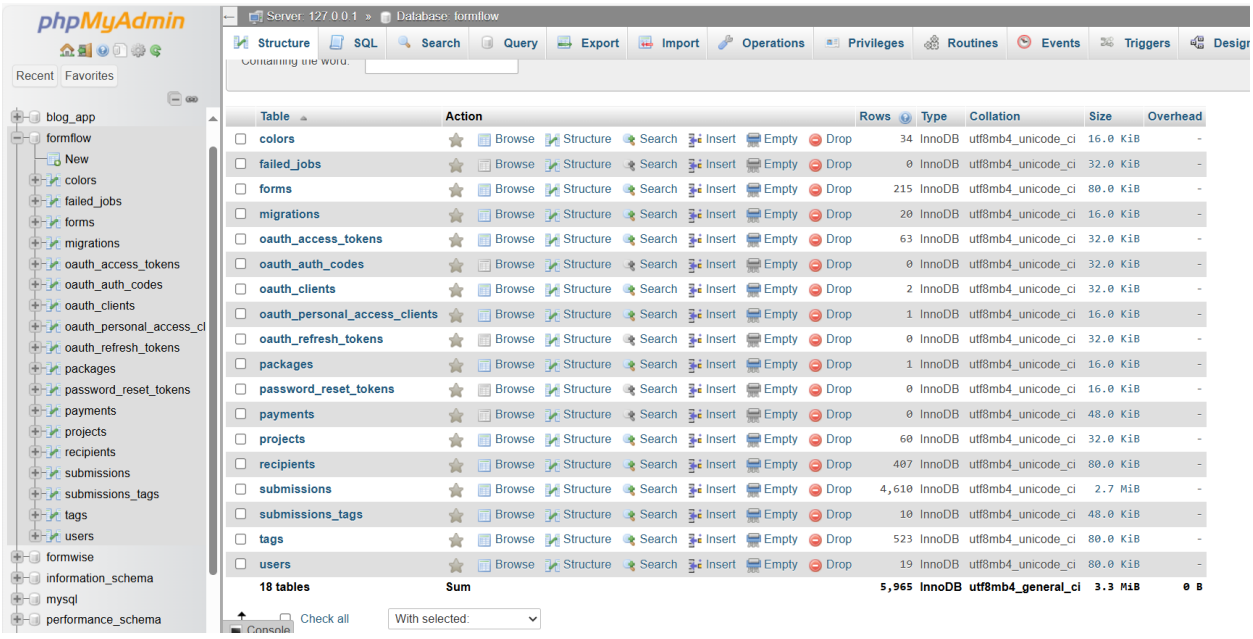
C. Од што се состои една веб апликација
Споменавме дека веб апликацијата е всушност напредна веб страница, односно веб страница која нуди интерактивност и подобро корисничко искуство на крајните корисници.
За една веб апликација да биде функционална и атрактивна потребно е да се обрати внимание на повеќе фактори. Идеално е апликацијата да има привлечен, интуитивен изглед, а притоа непречено да ги извршува функциите за кои е наменета.
Со оваа цел, за апликацијата FormFlow се изработени backend, frontend и база на податоци. Во продолжение ќе посветиме повеќе внимание на сите споменати.

База на податоци

База на податоци е збирка или колекција на податоци кои опишуваат одреден елемент. Со цел комуникација со базите на податоци, земање и анализирање на податоци, постојат системи за управување со бази на податоци (DBMSs). Овозможуваат создавање, дефинирање, ажурирање, пребарување и управување со базите. Познати вакви системи се MySQL, PostgreSQL, Oracle, Microsoft SQL Server итн.

За потребите на оваа апликација е искористено MySQL.

За полесен пристап до базата на податоци и информациите во неа е употребено phpMyAdmin и browser.



Слика 1 - Приказ на табелите од базата на податоци во phpmyadmin

Backend

Backend е делот од една веб-апликација кој работи во позадина и е одговорен за обработка на податоци, логика на апликацијата, и комуникација со базата на податоци. Тој не е видлив за корисниците, но е клучен за правилното функционирање на апликацијата. Бекендот ги обработува барањата од корисничкиот интерфејс (фронт-енд) и испраќа назад информации кои се прикажуваат на корисникот. За изработка на бекенд делот во FormFlow е искористен Laravel.

Што е Laravel?

Laravel е популарен PHP фрејмворк за развој на бекенд делот од веб-апликации. Тој се користи поради својата едноставност, брзина и моќни функции. Laravel нуди структуриран начин на организирање на кодот, што го прави развојот побрз и поефикасен. Нуди различни алатки и библиотеки кои помагаат во автоматизирање на чести задачи, како што се рутирање на барања, автентикација на корисници, управување со бази на податоци.

Предности на користење на Laravel во апликацијата

Користењето на Laravel во FormFlow, носи многу предности. Laravel ја поедноставува интеграцијата на автентикација и безбедност преку алатки како Laravel Passport, што овозможува безбедно управување со кориснички пристап и заштита на податоците. Потоа, Laravel е многу скалабилен, што значи дека може да се користи за развој на мали апликации, но и за големи и комплексни системи. Исто така има активна заедница и богата документација, што го олеснува процесот на учење и решавање на проблеми. Со користење на Laravel, апликацијата станува посигурна, полесна за одржување, и побрза за развој.

Работа со Laravel

За да се креира нов Laravel проект се користи командата

```
laravel new formflow-backend
```

или ако се користи Composer

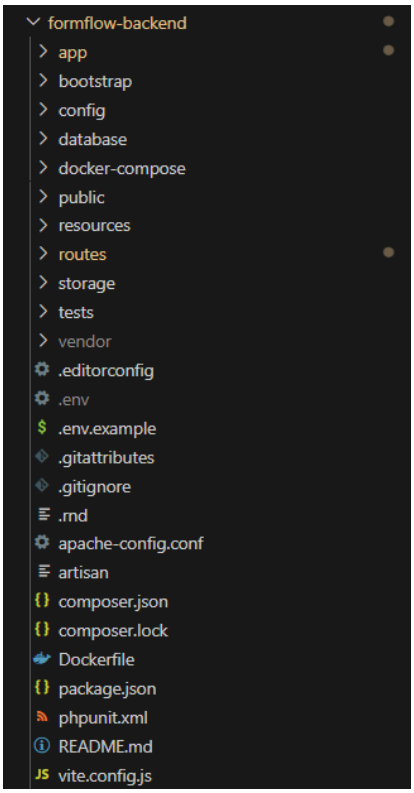
```
composer create-project laravel/laravel formflow-backend
```

Со ова, ќе се креира нова Laravel апликација во директориумот formflow-backend.

Структура

Откако ќе се креира проектот, структурата е организирана на следниот начин:

- app/: Содржи апликациски код, модели, контролери, и други класи.
- bootstrap/: Содржи код за почетно подигање на апликацијата.
- config/: Содржи конфигурациски датотеки.
- database/: Содржи миграции, factories, seeders.
- public/: Јавниот директориум од каде што се служат датотеките.
- resources/: Содржи извори на податоци како што се views, JavaScript и CSS.
- routes/: Содржи сите дефинирани рути на апликацијата.
- storage/: Содржи кеширани датотеки, логови, и сесии.
- tests/: Содржи тестови за апликацијата.
- vendor/: Содржи библиотеки инсталирани преку Composer.



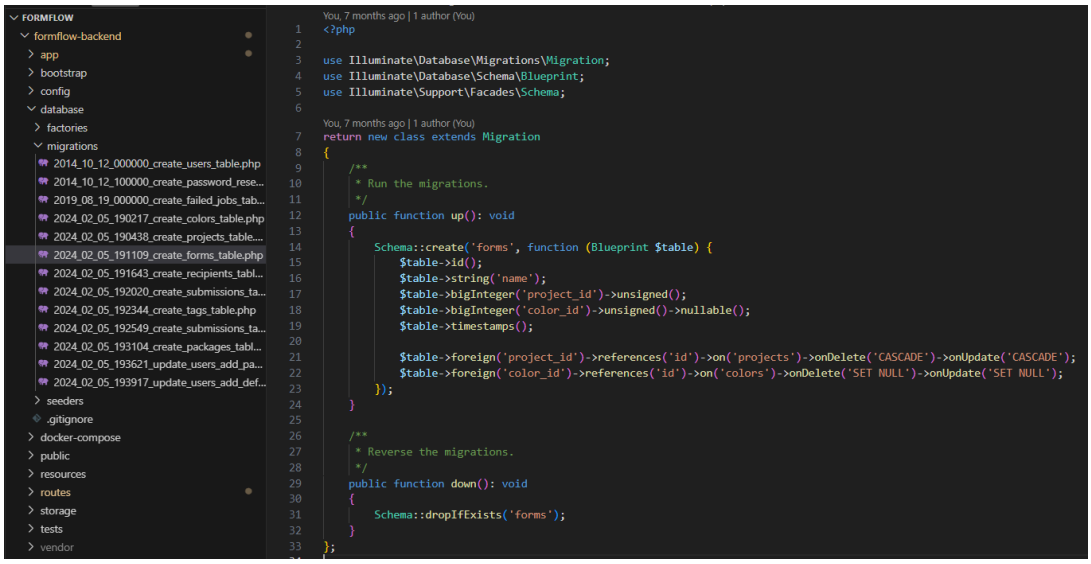
Работа со база и податоци во Laravel

Миграциите се начин за управување со базите на податоци во Laravel. За креирање на нова миграција се користи командата

```
php artisan make:migration create_users_table
```

Ова ќе креира нова миграција во папката ‘database/migrations’. Во миграцијата се дефинира структура на табелите во база. По дефинирањето на миграциите, со цел да се применат промените во база се извршува командата

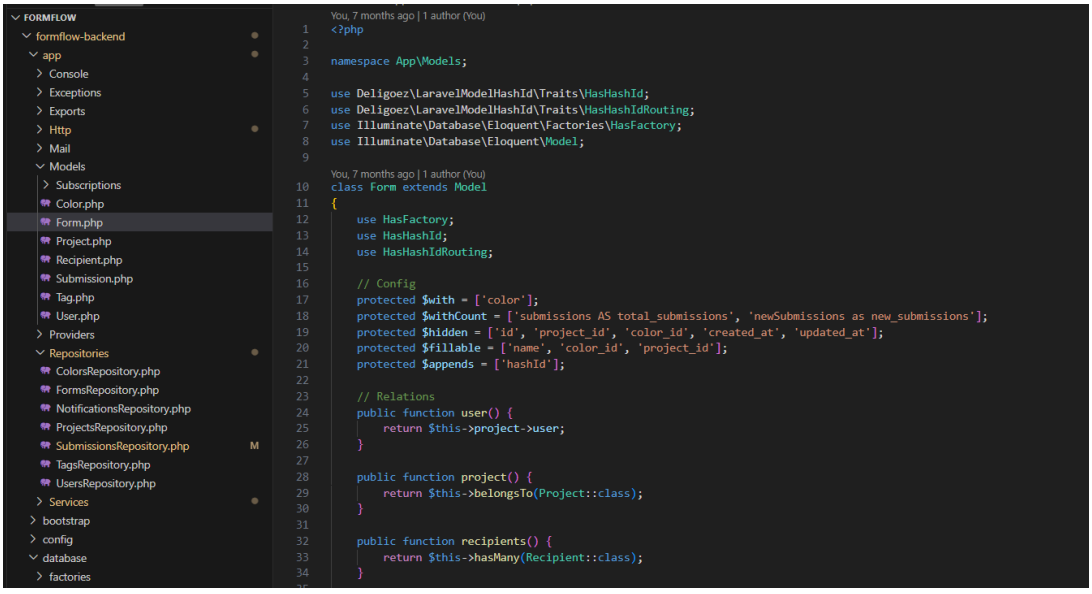
```
php artisan migrate
```



Слика 2 - Пример миграција за креирање на табелата за форми

Eloquent

Eloquent е ORM (Object-Relational Mapping) кој го користи Laravel за работа со бази на податоци. Со Eloquent може да се работи со базите на податоци користејќи PHP објекти наместо SQL команди. Eloquent моделите се всушност класи кои ги претставуваат табелите од база и обезбедуваат методи за работа со податоците. Eloquent може да се користи за креирање, читање, ажурирање и бришење на записи во базата.



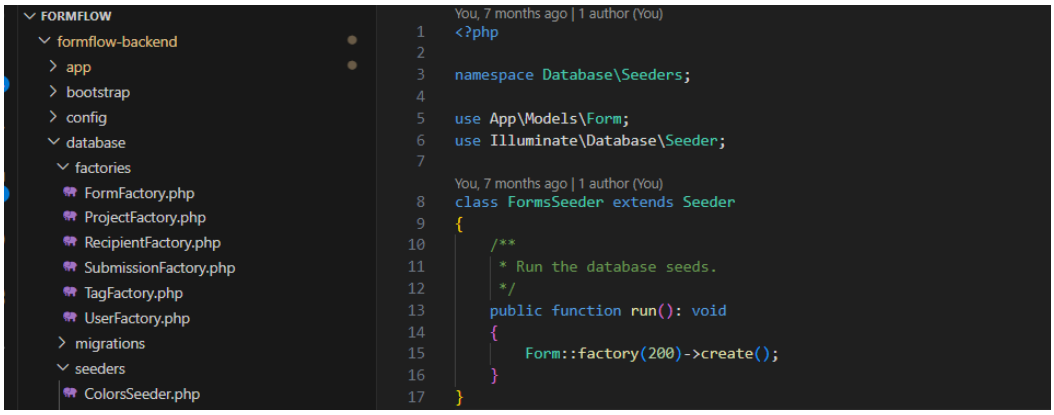
Слика 3 - Модел за ентитетот форма, приказ на неколку релации

Database seeders

Откако ќе се дефинираат миграциите и ќе се дефинираат табелите во база, можеме да креираме seeders – класи кои дозволуваат автоматско внесување на податоци во базата на податоци. Со ова, на едноставен начин, ја полниме базата со иницијални или тест податоци.

Нова seeder класа во фолдерот ‘database/seeders’ креираме, и ги извршуваме истите преку командите

```
php artisan make:seeder FormsTableSeeder
php artisan db:seed
```

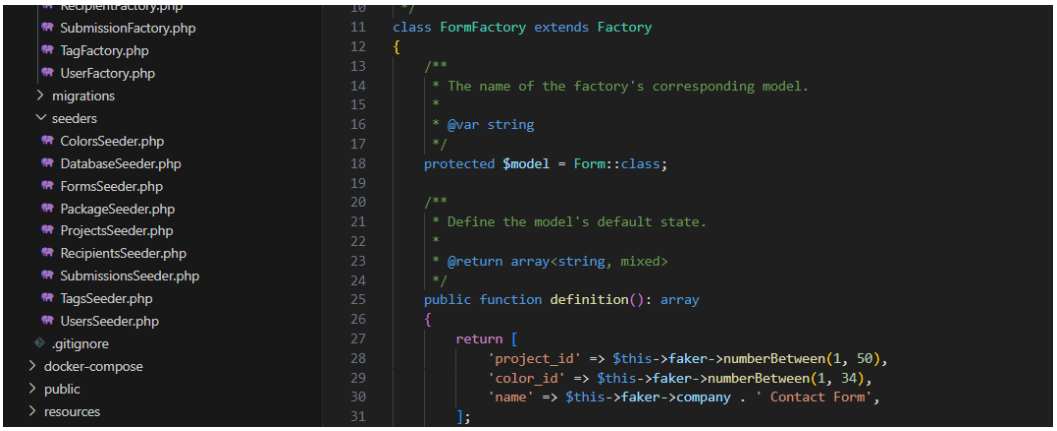


Слика 4 - Seeder за ентитетот Form

Database factories

Со цел да пополниме податоци во базата на податоци, потребно е претходно да ги дефинираме податоците. Factories се класи кои овозможуваат брзо креирање на објекти според даден модел, за тестирање и полнење на податоци. Factory класа се креира со

```
php artisan make:factory FormFactory --model=Form
```

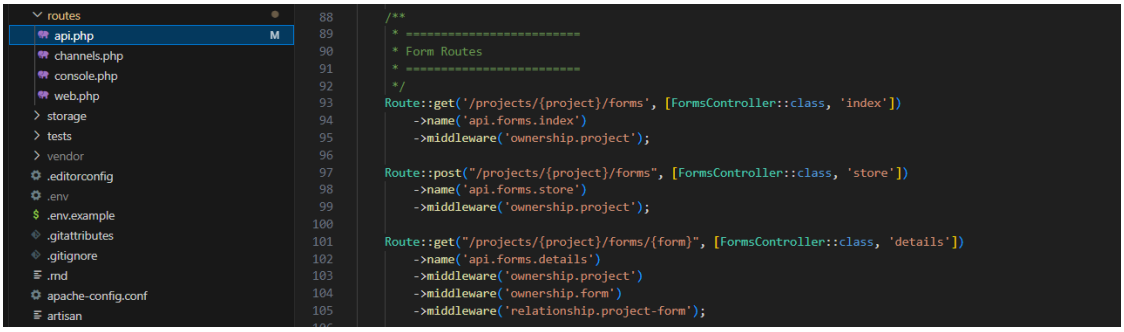


Слика 5 - Factory класа за моделот Form

Дефинирање на рути за API

За да може фронт-енд делот од апликацијата да комуницира со бекенд и базата, потребно е да се креираат API ендпоинти.

Во Laravel, рутите се дефинираат во ‘routes’ папката. Рутите за API обично се наоѓаат во ‘routes/api.php’. Овие рути се користат за целосната комуникација, како и за управување со CRUD операции (create, read, update, delete).

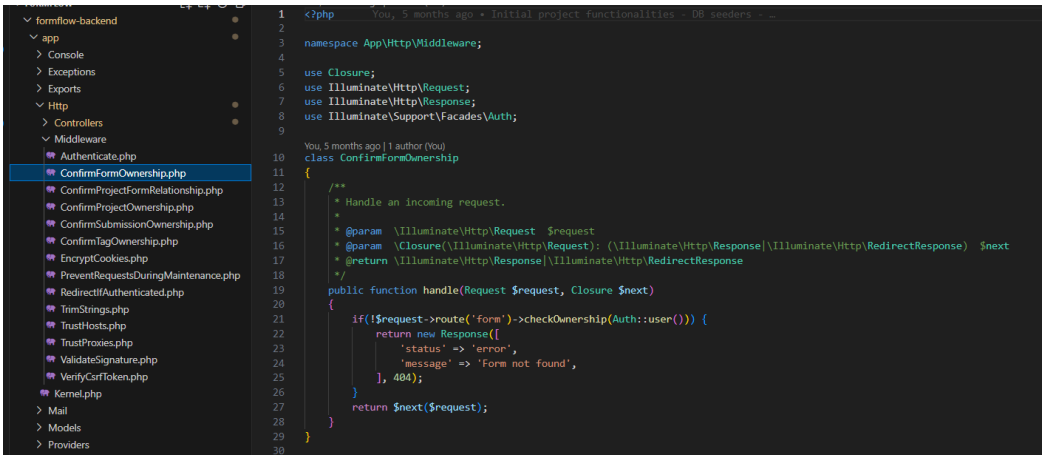


Слика 6 - Рути за forms ентитетот

Middleware

Middleware во Laravel се класи кои ја обработуваат HTTP барањето пред да стигне до контролерот. Тие се користат за да извршат различни проверки или операции, како што се автентикација, валидација на податоци, проверка на кориснички дозволи и многу повеќе.

Во случајов кај рутите на форма имаме дефинирано три middlewares за спроведување на проверки. Два од нив се однесуваат на ownership, односно проверуваме дали соодветниот проект и форма се во сопственост на моменталниот корисник, а третиот се однесува на проверка дали постои врска меѓу формата и проектот.



Слика 7 - Пример middleware за проверка на сопственост на форма

Структура на проект

Модели

Моделите во Laravel ги претставуваат табелите во базата на податоци и овозможуваат интеракција со податоците преку Eloquent ORM. Тие се наоѓаат во app/Models.

Слика 8 - Модел за Form

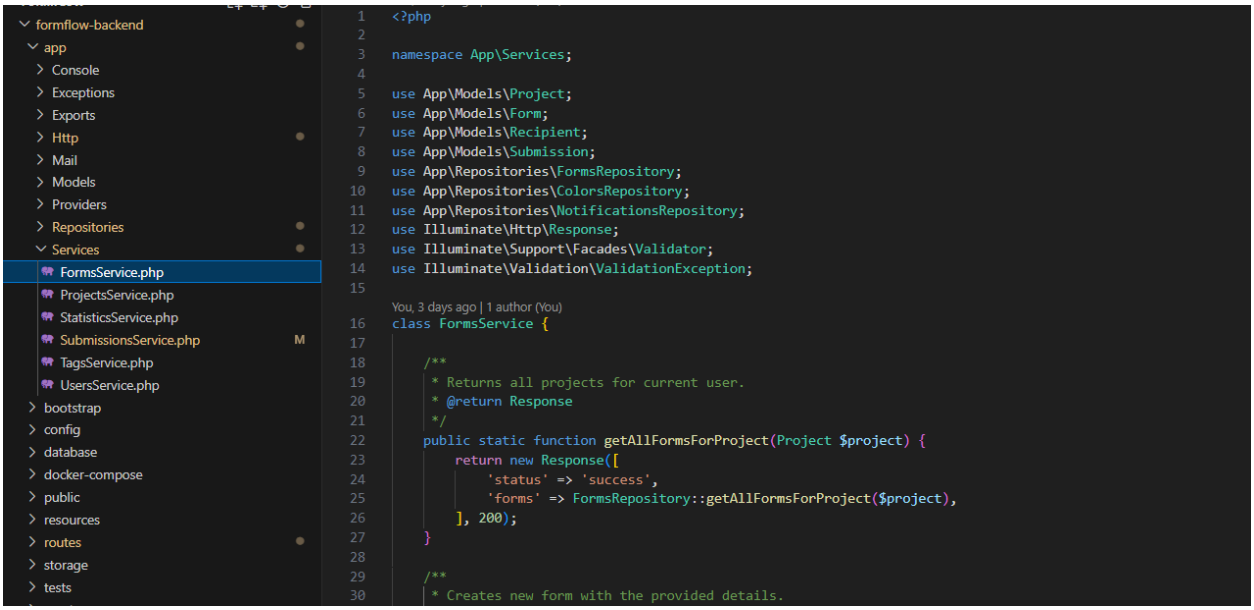
Контролери

Контролерите се класи кои ги обработуваат HTTP барањата и ги враќаат одговорите. Тие обично се наоѓаат во app/Http/Controllers.

Слика 9 - Контролер за Form

Сервиси

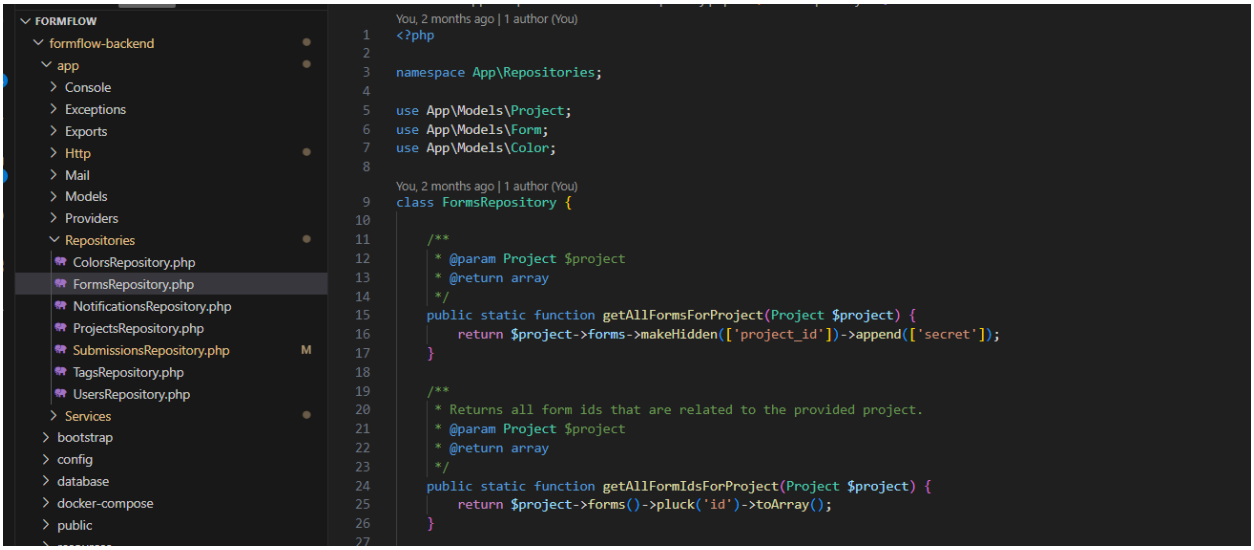
Сервисите се класи кои содржат бизнис логика која е одделена од контролерите за да се подобри организацијата и одржувањето на кодот. Тие се наоѓаат во app/Services.



Слика 10 - Сервис за Form

Репозиториуми

Репозиториумите се класи кои управуваат со директен пристап до податоците. Тие овозможуваат изолација на слојот за пристап до податоците и се наоѓаат во app/Repositories.



Слика 11 - Репозиториум за Form

Како се обработува барање во Laravel

Кога ќе се испрати барање до Laravel апликацијата, се случува следново:

1. **Рутата се препознава:** Laravel ја проверува рутата што одговара на URL-то на барањето (во routes/api.php).
2. **Контролерот се повикува:** Рутата ја повикува соодветната метода од контролерот. На пример, ако барањето е за /forms со GET метод, ќе се повика index() метода од FormController.
3. **Сервисот го обработува барањето:** Ако има сервис, контролерот го користи сервисот за да ја обработи бизнис логиката (на пр. преземање на сите форми).
4. **Моделите и Репозиториумите ја вршат интеракцијата со базата:** Ако се користат репозиториуми, сервисот или контролерот комуницира со базата преку нив за да ги добие или зачува потребните податоци.
5. **Одговорот се враќа:** Контролерот го враќа одговорот (на пример, JSON).

Frontend

Frontend е делот од веб-апликацијата кој е видлив и интерактивен за корисниците. Тоа е корисничкиот интерфејс (UI) кој корисниците го користат за да комуницираат со апликацијата. Frontend-от е одговорен за прикажување на податоците добиени од backend-от и за прифаќање на кориснички внес.

Frontend делот на **FormFlow** е изработен со користење на Vue.js 3.

Што е Vue.js 3?

Vue.js 3 е модерен JavaScript фрејмворк за развој на кориснички интерфејси и апликации од една страна (SPA). Тој е лесен за користење, реактивен, и овозможува брз и флексибилен развој.

Vue.js 3 користи компоненти, што овозможува креирање на повторно употребливи делови на кодот кои лесно се интегрираат и управуваат. Ова го прави развојот поорганизиран и полесен за одржување.

Во овој дел, ќе ги разгледаме основите на креирање на проект со Vue.js 3, дефинирање на рутирање, управување со состојба (state) користејќи Pinia, работа со репозиториуми, Axios за HTTP барања, компоненти, страници, настани.

Креирање на Vue.js 3 проект

За да креирате нов Vue.js 3 проект, можете да се користи Vue CLI или Vite, се извршуваат командите:

```
npm create vite@latest my-vue-app -- --template vue
```

```
cd my-vue-app
```

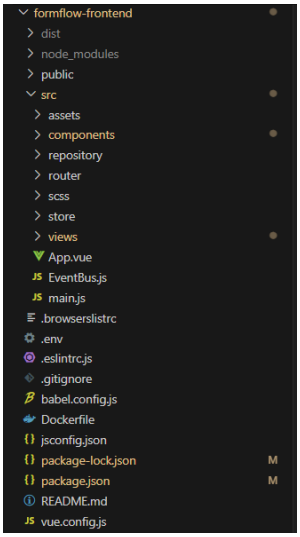
```
npm install
```

```
npm run dev
```

Структура

По креирање на проектот, се добива основна структура која се состои од повеќе елементи, и ги содржи сите потребни датотеки и документи за стартување на иницијален проект.

- node_modules - Ги содржи сите зависности на проектот
- public - Статички ресурси
- src - Изворни датотеки
- assets - Статички ресурси како слики
- components - Vue компоненти
- views - Страници на апликацијата
- App.vue - Главна апликациска компонента
- main.js - Влезна точка за апликацијата
- .gitignore - Ги исклучува датотеките од Git
- index.html - Главна HTML датотека
- package.json - Информации за проектот и зависности



Стартување и пристап до Vue проектот

По креирањето на проектот, потребно е прво да се инсталираат сите зависности, односно да се изврши командата

```
npm install
```

Потоа, за да се стартува проектот, со што истиот ќе може да се пристапи на browser, се извршува

```
npm run dev
```

Рутирање во Vue

Рутирањето е клучен концепт во развојот на апликации со една страна (SPA) со Vue.js 3. апликациите користат единствена HTML страница и динамички ги вчитуваат и менуваат компонентите, без потреба од целосно освежување на страницата.

За да се постигне ова, потребно е да се има систем кој управува со различните патеки (URL адреси) во рамките на апликацијата. Рутирањето овозможува навигација помеѓу различни компоненти или "страници" на апликацијата, користејќи го притоа истиот HTML документ.

Пример, во апликација како FormFlow, токму рутирањето овозможува имање на различни страници за прикажување на сите форми, преглед на поединечни одговори, менаџирање на проекти итн.

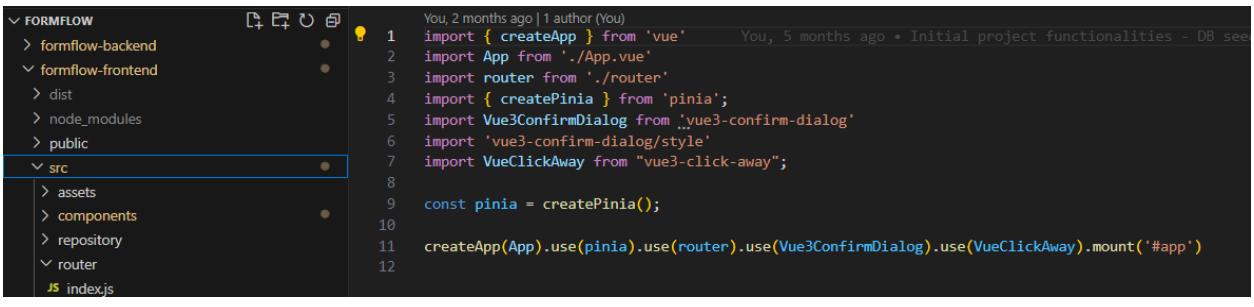
Во Vue.js 3, рутирањето се имплементира со помош на **Vue Router**, кој се инсталира со помош на командата:

```
npm install vue-router
```

Потоа, се креира посебна датотека за дефинирање на сите рути, најчесто src/router/index.js. каде се дефинираат сите рути кои апликацијата ќе ги користи.

Слика 12 - Дефиниција на рути

По дефинирањето на рутерот, потребно е истиот да се интегрира во проектот. Во main.js датотеката. И потоа со router-link се поставуваат линковите за рутирање.



Слика 13 – Интеграција на vue router

```
25              <li class="nav-item">
26                  <router-link to="/submissions" class="nav-link" exact>
27                      <img src="@/assets/icons/submissions.svg" alt="Dashboard">
28                      <span>Submissions</span>
29                  </router-link>
30              </li>
31
32              <li class="nav-item">
33                  <router-link to="/tags" class="nav-link" exact>
34                      <img src="@/assets/icons/tags.svg" alt="Dashboard">
35                      <span>Tags</span>
36                  </router-link>
37              </li>
```

Слика 14 – Употреба на vue router

Управување со состојба во Vue

Управување со состојба (state management) е процес на следење и управување со податоците што ги користи апликацијата. Во веб-апликациите, "состојба" се сите информации кои апликацијата ги чува и менува додека корисникот ја користи, како што се внесените податоци, статусот на формите, активниот корисник, и слично.

Управување со состојба е потребно за да се осигура дека сите делови од апликацијата имаат точни и актуелни податоци. На пример, ако корисникот се логира, состојбата ја чува информацијата за логираниот корисник и ја користи за да одлучи кои податоци и функционалности се достапни за него.

Pinia

Во Vue.js 3, Pinia е библиотека која помага да се управува со оваа состојба на еден централен и организиран начин, што го прави кодот полесен за разбирање и одржување.

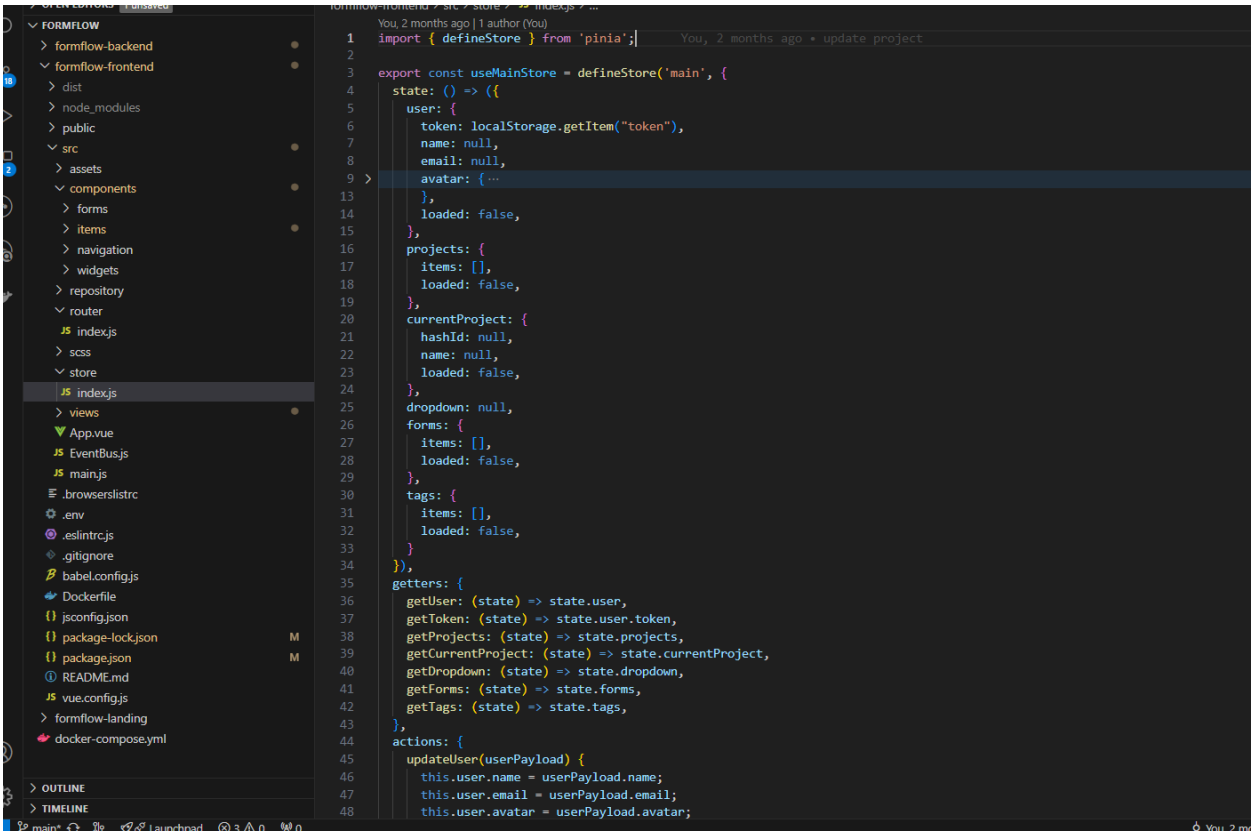
За инсталирање на Pinia се користи командата

`npm install pinia`

Откако ќе се инсталира, потребно е да се направи интеграција, слично како во примерот погоре во main.js документот.

По интеграцијата се креира Store со Pinia, каде што се дефинираат:

- **state**: Дефинира објект кој ги содржи сите податоци (состојби) кои ќе бидат користени во Store-от
- **getters**: Пресметани вредности кои овозможуваат да се пристапат податоците
- **actions**: Методите кои овозможуваат да се менува состојбата



Слика 15 – Store во Vue

Откако ќе биде дефиниран, истиот може да се вклучи во компонентите и да се користи со повикување на акциите од него.

Слика 16 – Употреба на store

Axios

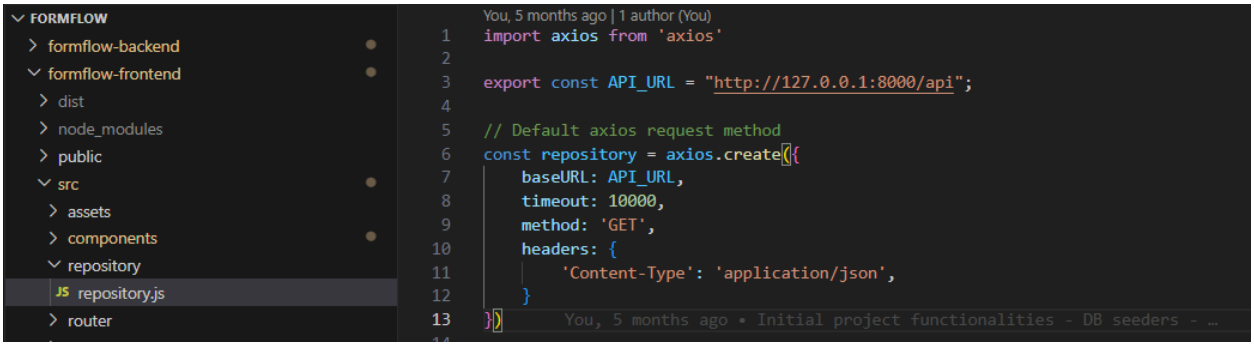
Axios е популарна JavaScript библиотека што се користи за правење HTTP барања од клиентската страна (фронт-енд) до серверската страна (бекенд). Со други зборови, Axios овозможува комуникација помеѓу фронт-ендот на апликацијата и серверот, каде што се наоѓа бекендот.

Ова е многу важно за веб-апликации, каде што корисниците внесуваат податоци и сакаат да ги испратат до серверот за обработка и чување, или пак сакаат да добијат информации од серверот за прикажување.

За да се користи axios потребно е најпрво да се инсталира со командата:

```
npm install axios
```

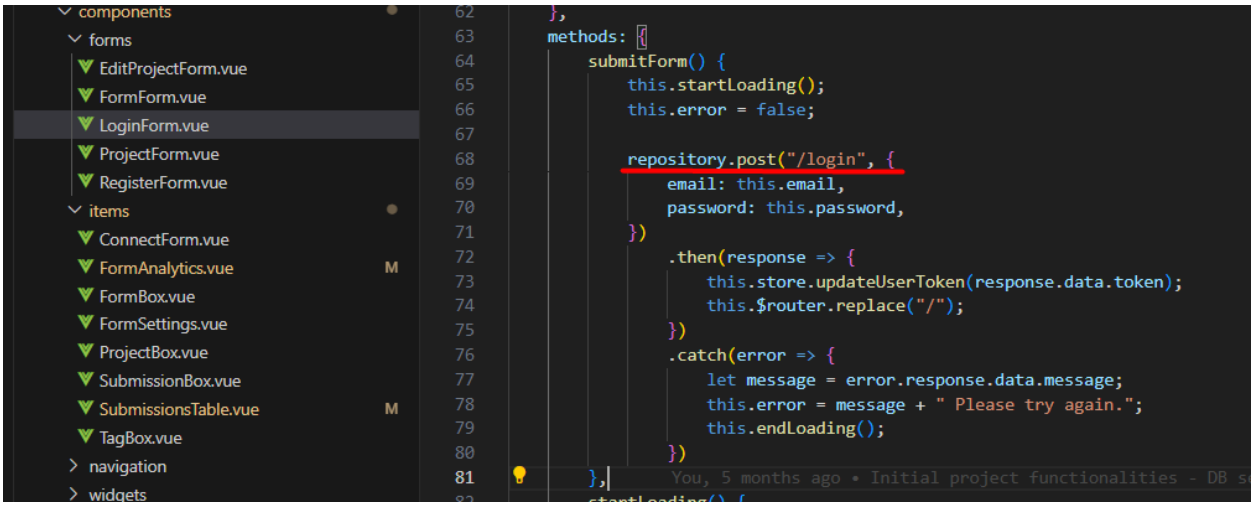
Откако ќе се инсталира, потребно е да се конфигурира во проектот. Овде се дефинираат основните параметри како API_URL со цел да не мора на секој повик да се пишува целиосниот url.



Слика 17 – Конфигурација на axios

Откако е завршена конфигурацијата, може да се испраќаат барања до бекендот и понатаму да се обработуваат добиените одговори.

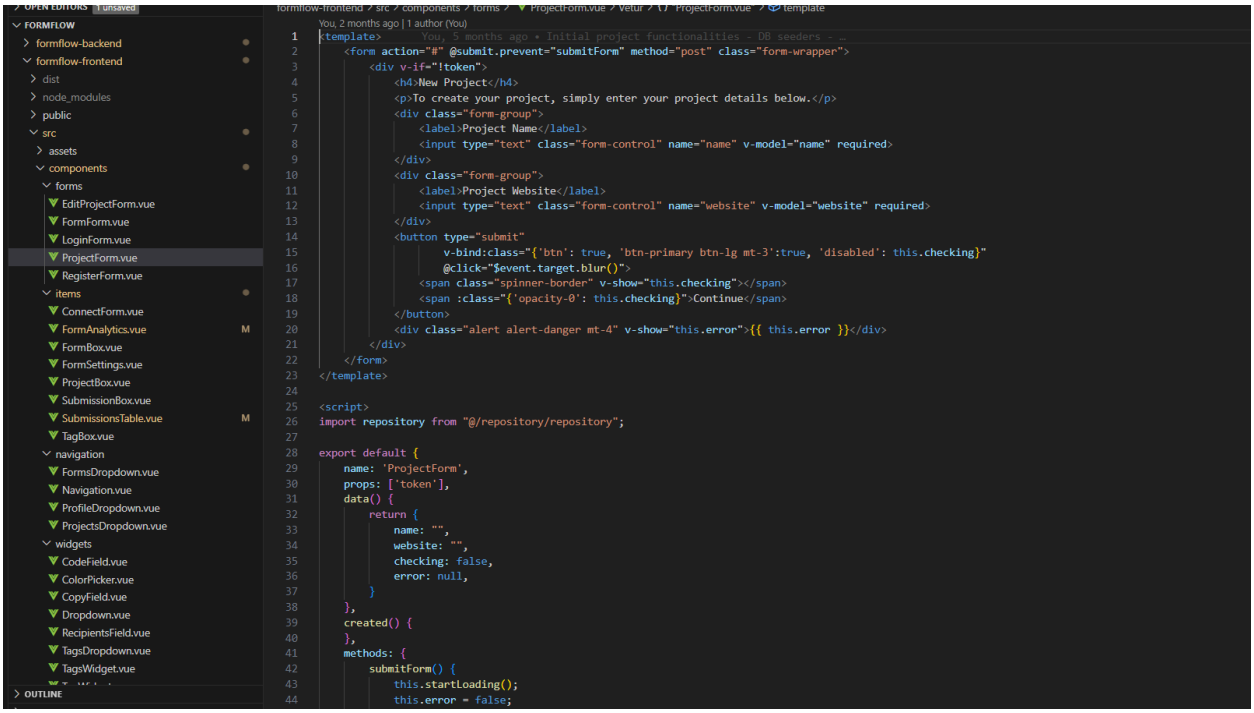
Слика 18 – Употреба на axios за правење на повици за читање до бекенд



Слика 19 – Употреба на axios за правење на повици за запишување до бекенд

Компоненти

Компоненти се затворен дел од код кој може да се реупотребува на различни делови низ апликацијата. Се состои од html, css и javascript кои се потребни за нејзино функционирање. Како компоненти може да се креираат најразлични елементи како форми, табели, копчиња, па дури и цели страници.



Слика 20 – Пример компонента

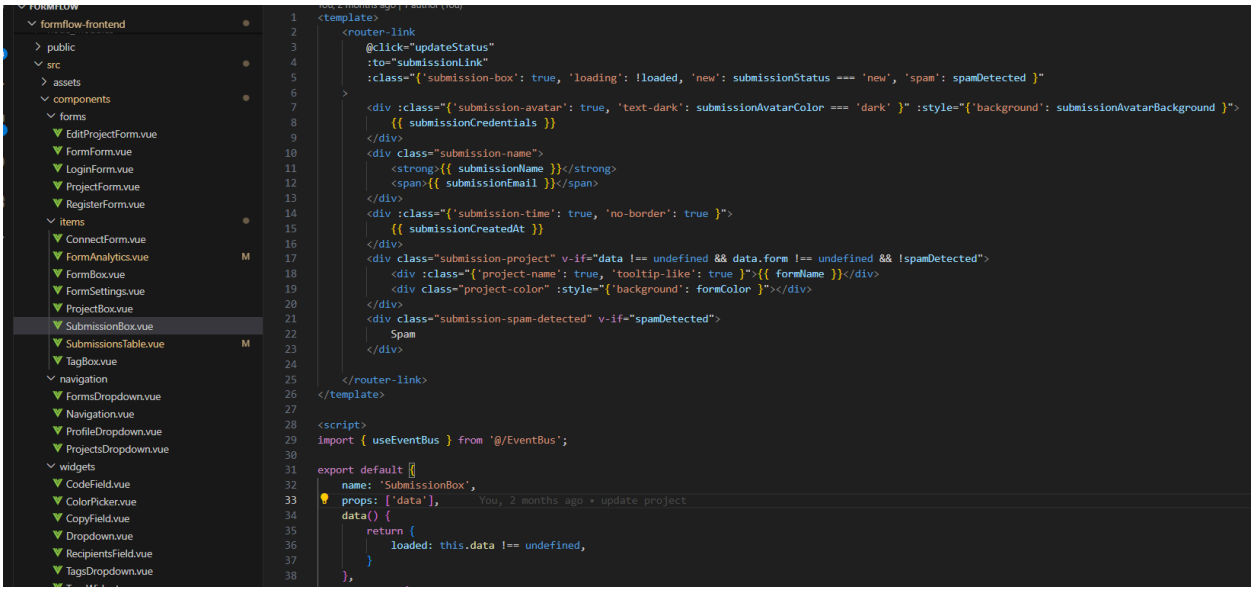
Во примеров е прикажана компонента за форма за креирање проект насловена ProjectForm, која понатаму е интегрирана во одредена страница и се прикажува според одредени услови. Компонентава целосно се справува со процесот на додавање на нов проект во апликацијата.

Во дел од компонентата во script областа може да се забележат повеќе елементи како name, props, data, methods итн.

Props

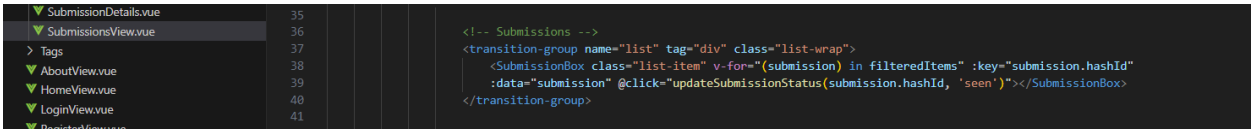
Props се кратенки за “properties” и се користат за пренесување на податоци од родител компонентата до компонентата дете. Ова овозможува компонентите да бидат пофлексибилни и да се употребуваат со различни вредности.

За комуникација пак помеѓу компоненти кои не се во релација родител – дете е искорестен event bus. Преку него се емитираат настани и се реагира за соодветни настани исто така. Ова овозможува повеќе компоненти да реагираат на исти настани, и да споделуваат податоци без да се испраќаат како props.



Слика 21 – Употреба на event bus и props

Во примеров како prop е испратен параметарот data, кој содржи податоци за еден одговор на форма. Тоа е искористено при листање на сите одговори, така што за секој одговор се користи истата компонента, но со различни податоци соодветно кои се испраќаат како prop со :data=”submission”.



Слика 22 – Испраќање на параметар како prop

Методи / Акции

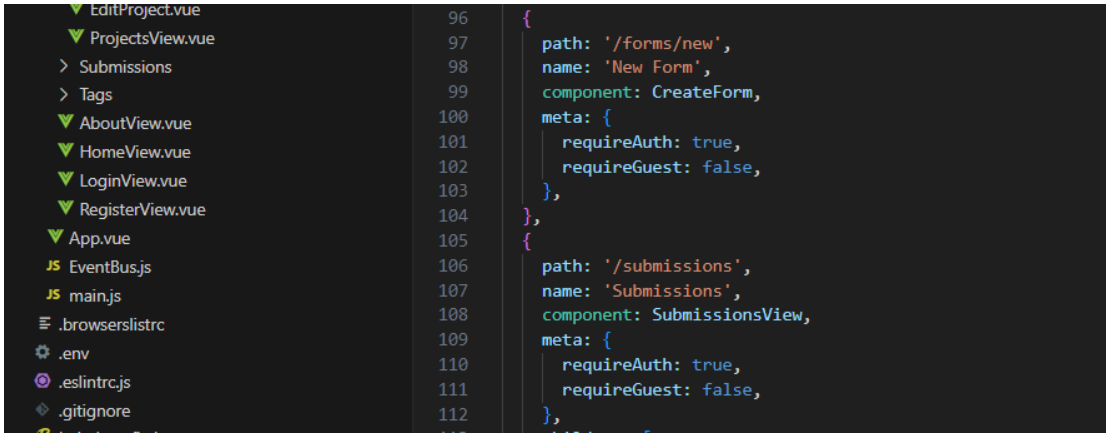
Методи во Vue.js се функции кои се дефинираат во скрипт делот на компонентата и може да се користат за обработка на податоци или за манипулирање со состојба. Тие обично се користат како реакција на настани (events), како што се кликови на копчиња или испраќање на форми.

Слика 23 – Приказ на дефинирани методи

Во примеров се неколку акции како читање и листање на таговите, додавање на одреден таг за одреден одговор, справување со клик при додавање на таг.

Страници

Страниците се специјални типови на компоненти кои претставуваат цели делови од апликацијата, обично врзани за рути (routes). На пример, во апликацијата имаме страници за почетен изглед, регистрација, најава, проекти, форми, одговори, тагови. Страниците се компоненти кои се вчитуваат врз основа на рутата дефинирана во Vue Router.

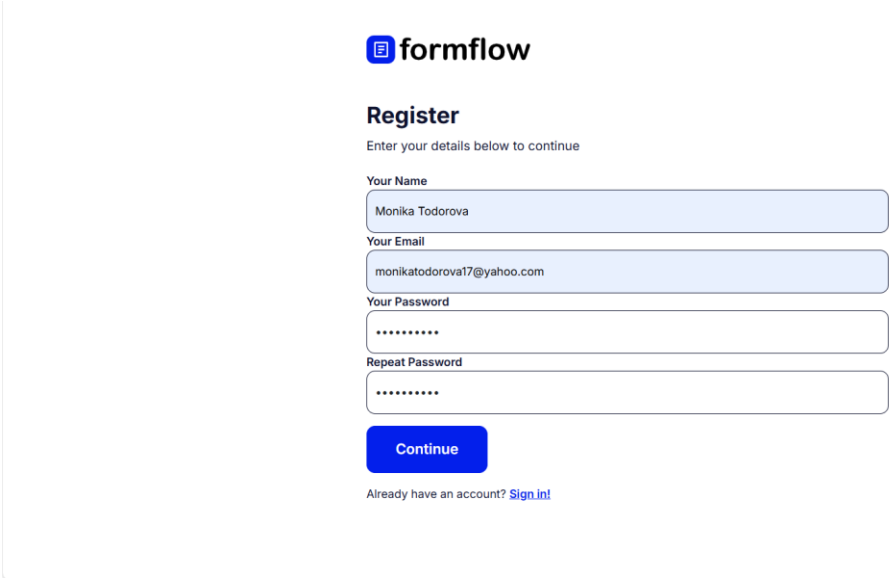


Слика 24 – Страници поврзани со рути

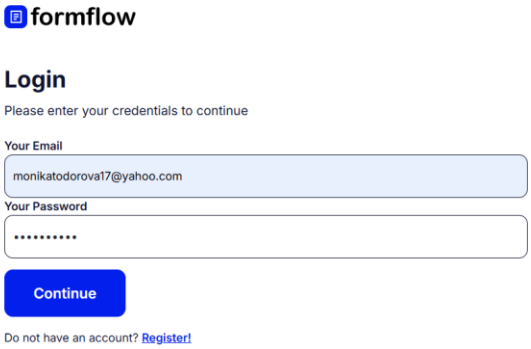
Преглед на апликацијата FormFlow

Креирање профил и најава

Апликацијата нуди едноставни и интуитивно дизајнирани екрани за креирање на корисички профил и најава.



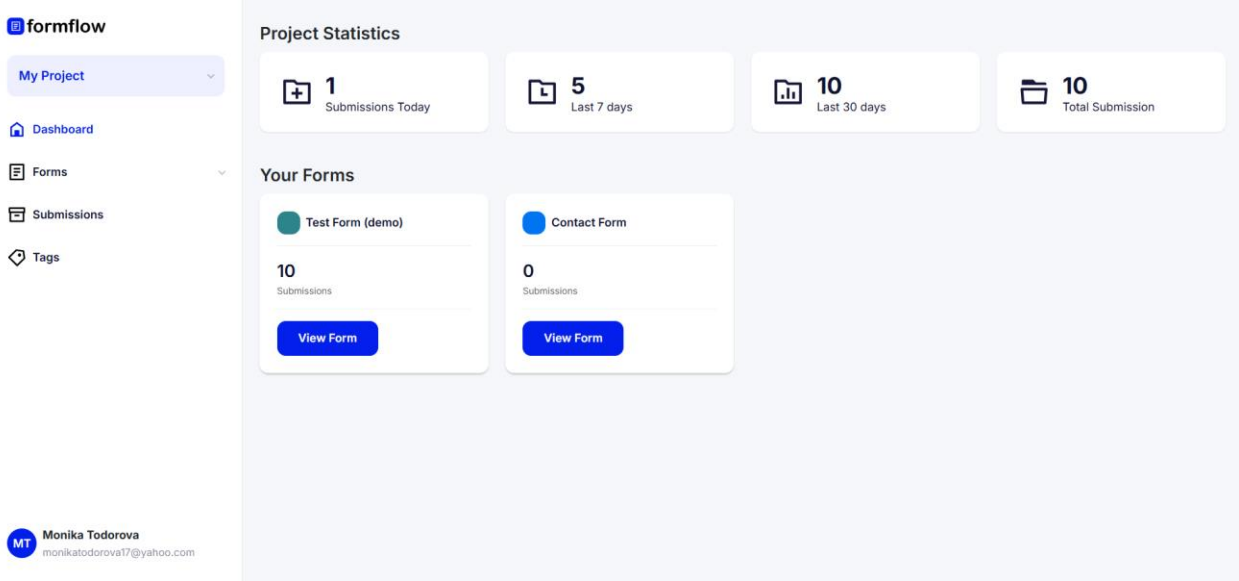
Слика 25 – Екран за креирање на профил



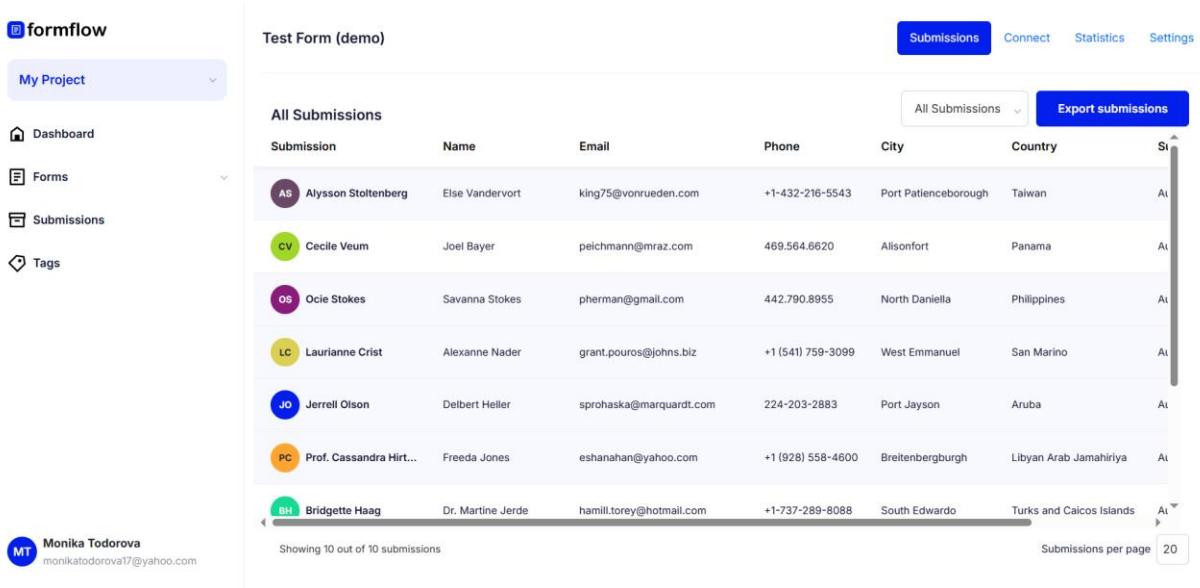
Слика 26 – Екран за најава

Иницијален проект и пример форма

Со креирање на профилот на корисникот се креираат и иницијален проект, кој се поставува како стандарден / селектиран. За истиот проект се креираат и две форми. Една од формите (Test form) содржи и пример одговори, со цел корисникот лесно да добие увид во тоа како неговите податоци ќе се прикажуваат во апликацијата. Другата форма е празна, со спремен код за интеграција, по што ќе може да добие одговори за неа.



Слика 27 – Иницијален екран



Слика 28 – Приказ на тест формата

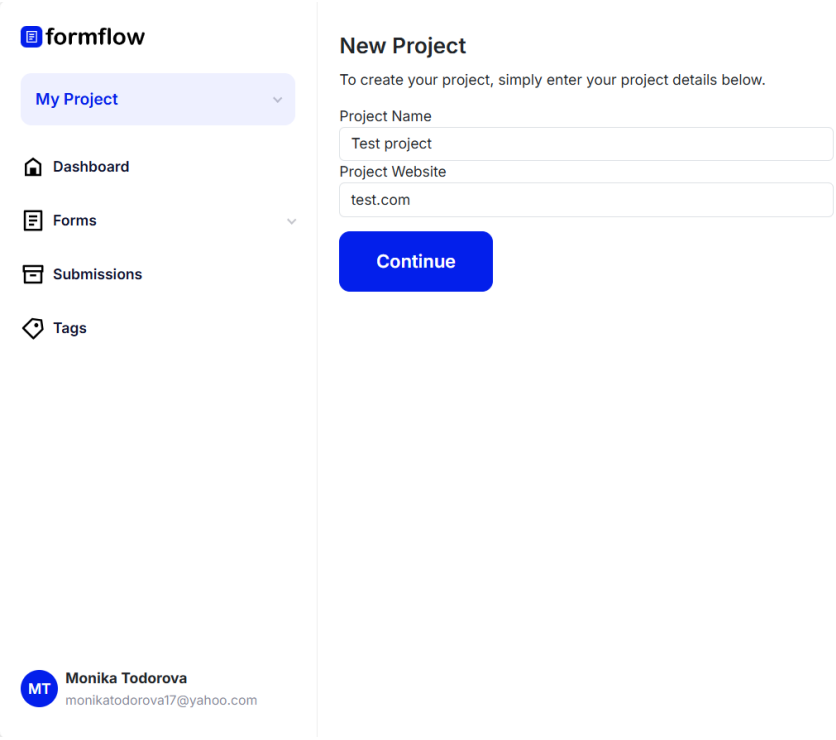
Креирање и менаџирање на проекти

Корисникот покрај иницијалниот проект може да креира и други проекти. Проектот всушност претставува опција за групирање на формите.

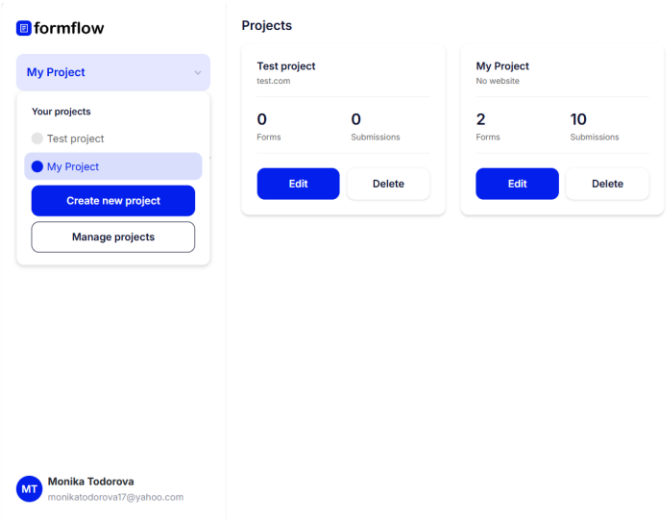
Идејата за групирање по проект би се користела на пример во случај кога корисникот има две различни веб страни во кои соодветно има форми, би можел да креира различен проект за секоја од страните, и во секој од проектите, да ги креира и конектира соодветните форми.

Креирањето на проект се одвива со внесување на информации во едноставна форма. По успешно креирање на новиот проект, тој се појавува во менито од проекти, како и во листата проекти наменета за нивно менаџирање. Секој од проектите може соодветно да се едитира и / или избрише.

Доколку селектираме некој од излистаните проекти во менито од навигацијата, истиот ќе се постави како главен / селектиран проект, и останатите податоци ќе се ажурираат соодветно на тоа.



Слика 29 – Екран за креирање на нов проект

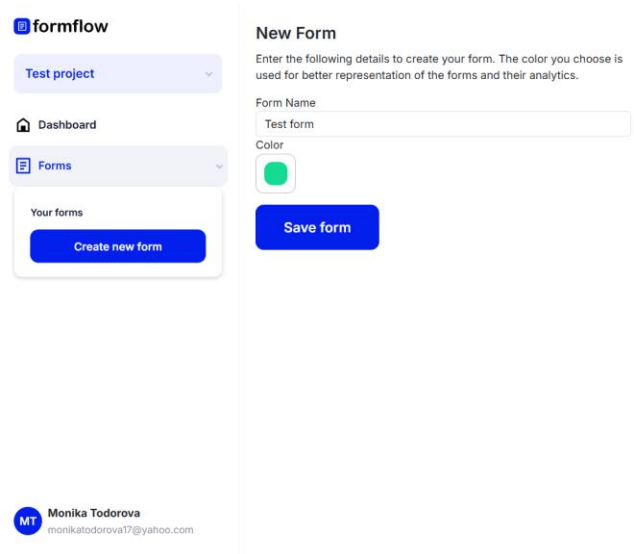


Слика 30 – Екран за приказ на креираните проекти

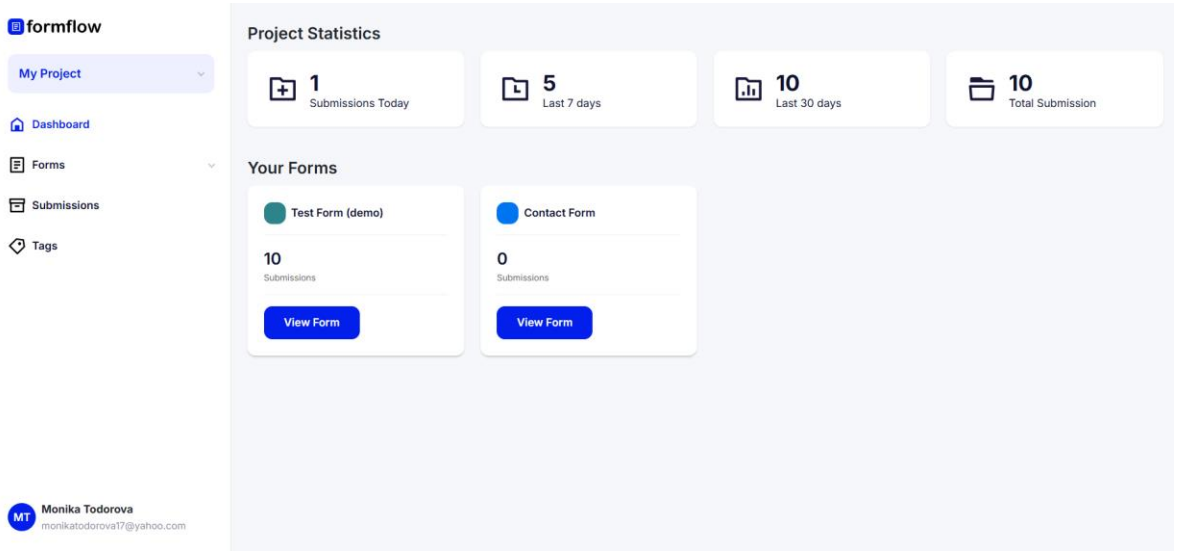
Креирање и менаџирање на форми

Покрај креирањето на проекти, корисникот може да креира и нови форми. Формата за креирање на нови форми е достапна преку избирање на опцијата Create new form во менито на форми во навигацијата.

За да се креира една форма, потребно е само да се внесе нејзино име и може да се селектира боја, со единствена цел полесно распознавање на формата и понатаму одговорите поврзани со неа.



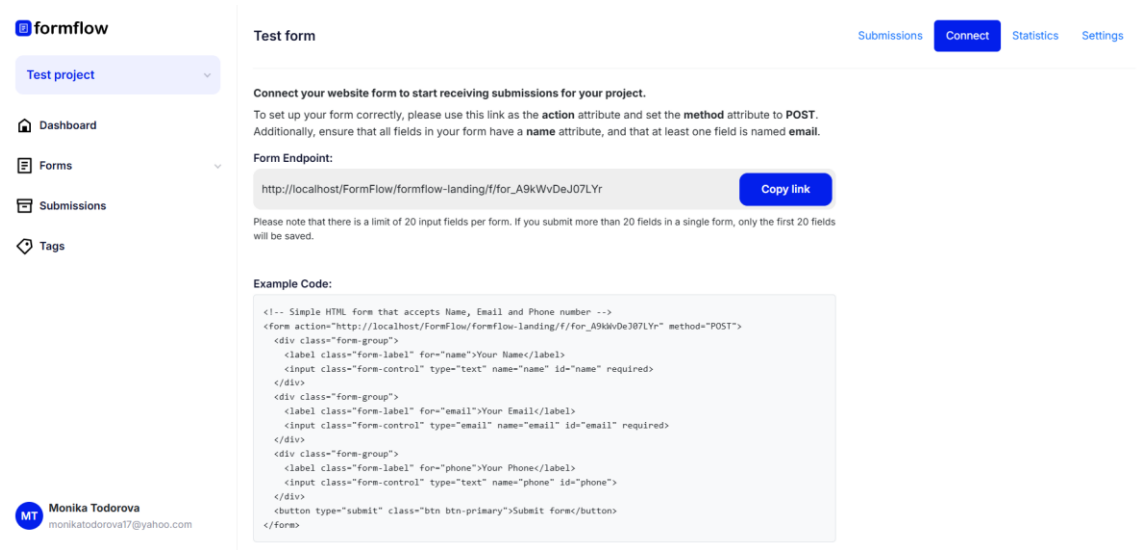
Слика 31 – Екран за креирање на нова форма



Слика 32 – Екран за приказ на креираните форми

За новокреираната форма да може да добива одговори поврзани со неа во апликацијата, потребно е истата да се конектира. За тоа е креиран посебен екран кој ги содржи интрукциите за таа акција. Имено во моменталната верзија на апликацијата, за формата да се конектира потребно е во action полето на формата од веб страната на корисникот да се внесе дадениот endpoint. Со тоа, формата ќе биде поврзана во апликацијата и одговорите од неа ќе се зачувуваат и прикажуваат овде.

Дополнително на екранов е дадена пример форма со соодветно поврзан ендпоинт која може да се искористи за креирање на минимална функционална онлајн форма.



Слика 33 – Екран со инструкции за конектирање на формата

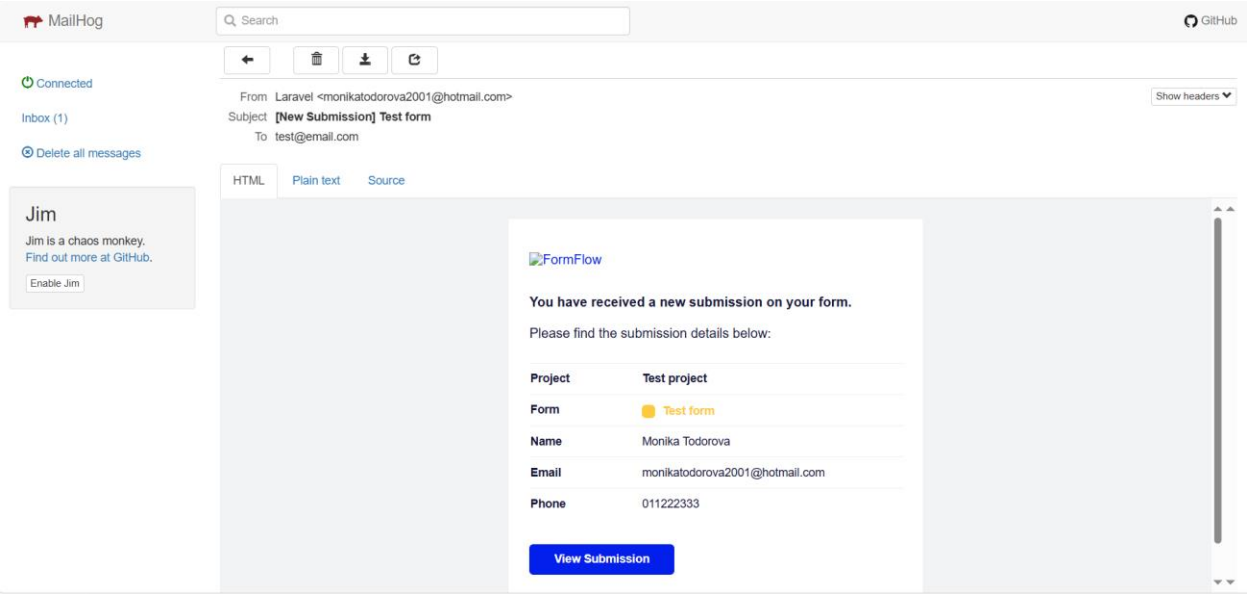
Слика 34 – Приказ на пример формата

Прибирање и менаџирање на одговори

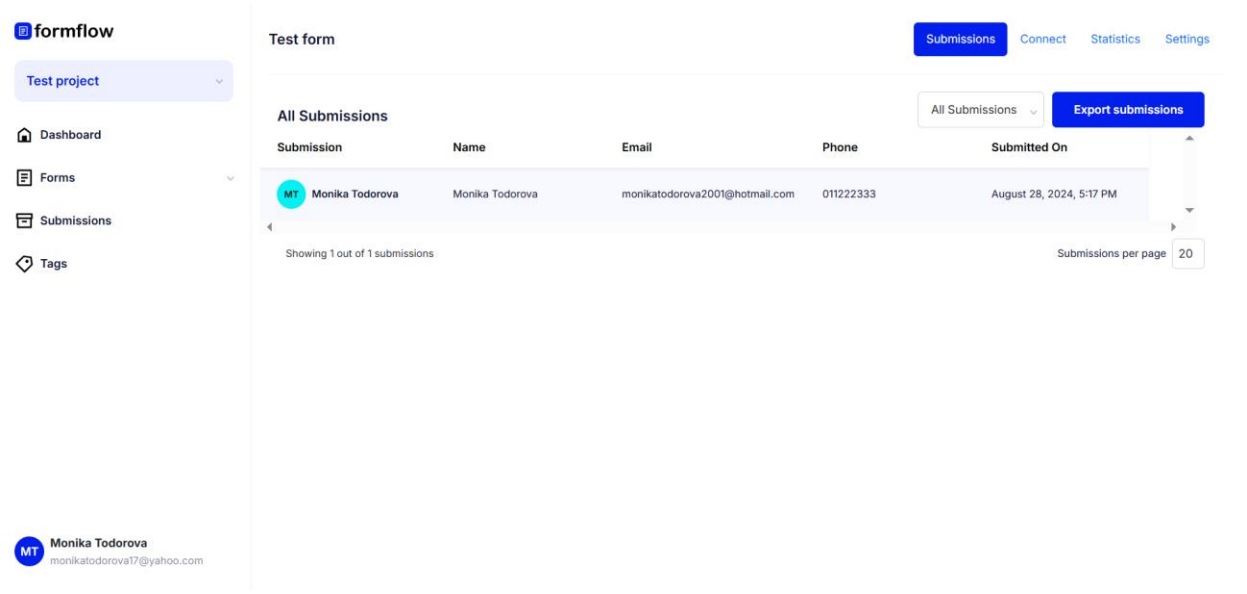
Откако корисникот ќе ја поврзе својата онлајн форма со ендпоинтот од апликацијата. Сите одговори кои ќе бидат испратени таму ќе пристигаат во апликацијата. По испраќањето на одговорот, на соодветната страна се покажува приказ за успешно испраќање.

Слика 35 – Успешно испраќање на одговор преку креираната форма

Во апликацијата, во главниот екран за формата може веднаш да се види новата порака. Притоа се испраќа и нотификација на емаил за добивањето на новиот одговор од формата. Локално тоа е истестирано со MailHog.

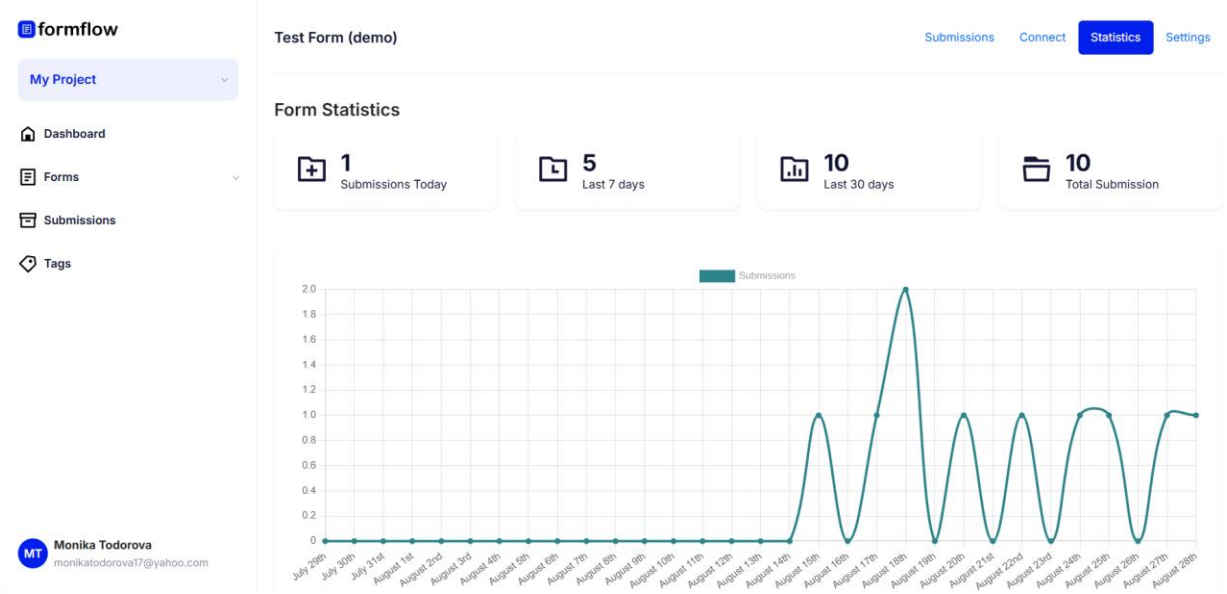


Слика 36 – Емаил известување за новодобиениот одговор



Слика 37 – Приказ на новодобиениот одговор во апликацијата

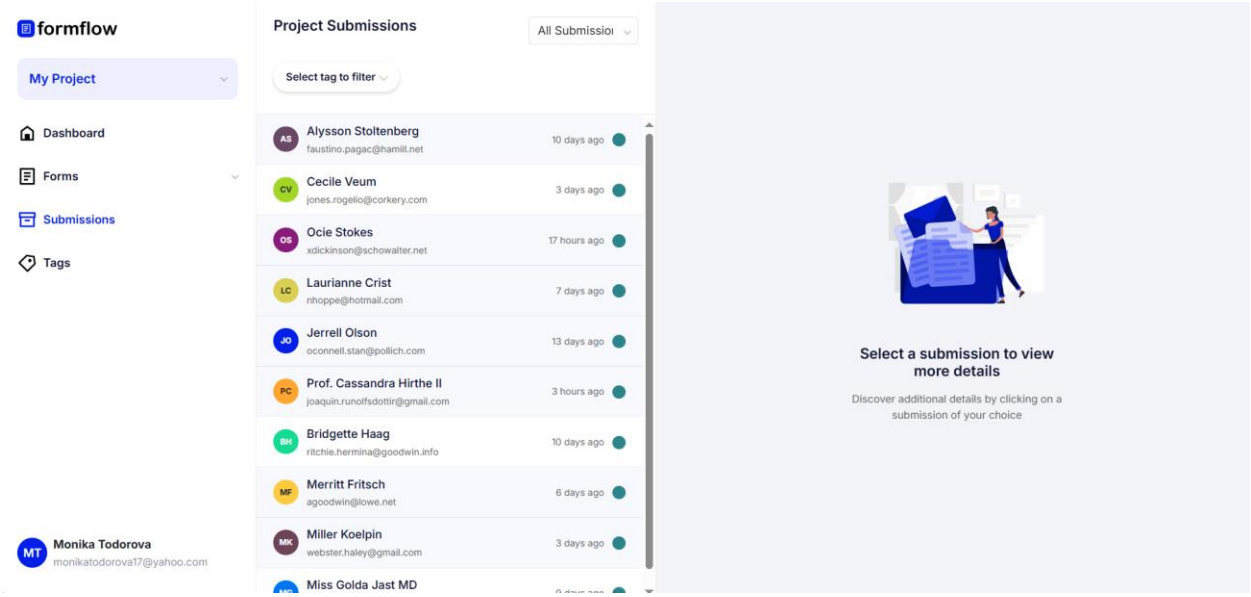
За секоја од формите е обезбеден дел со аналитика каде што може да се види по текот на претходниот месец, по ниво на денови, колкав е бројот на добиени одговори.



Слика 38 – Приказ на екран со аналитики

Приказ на одговори и менаџирање со тагови и статус

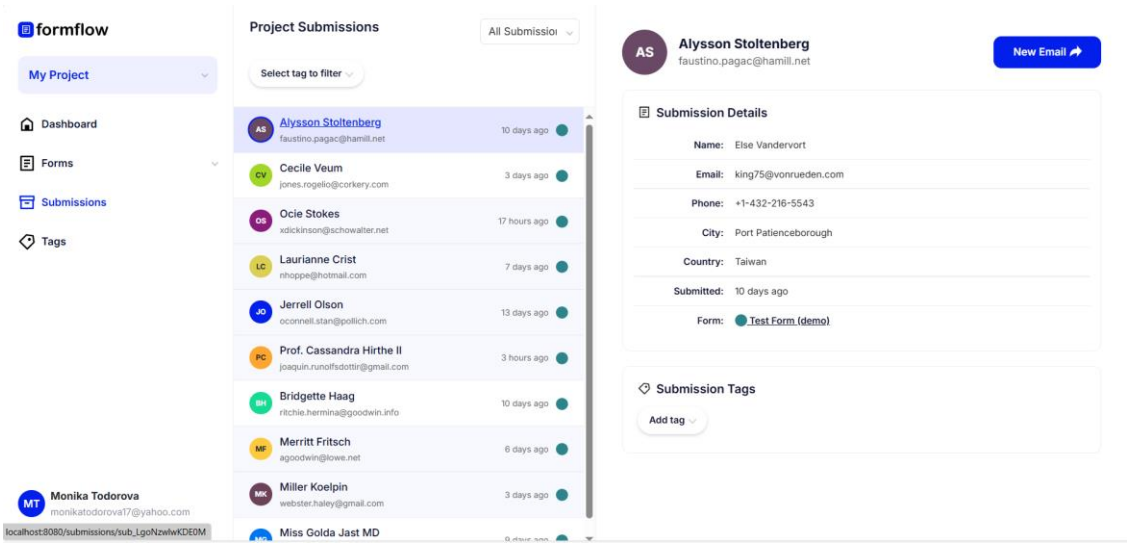
Освен приказот на одговорите поврзани со една форма, овозможен е и приказ на одговорите поврзани со целиот проект. Во делот на submissions се излистани сите одговори кои се поврзани со некоја од формите во моментално селектираниот проект.



Слика 39 – Приказ на сите одговори од даден проект

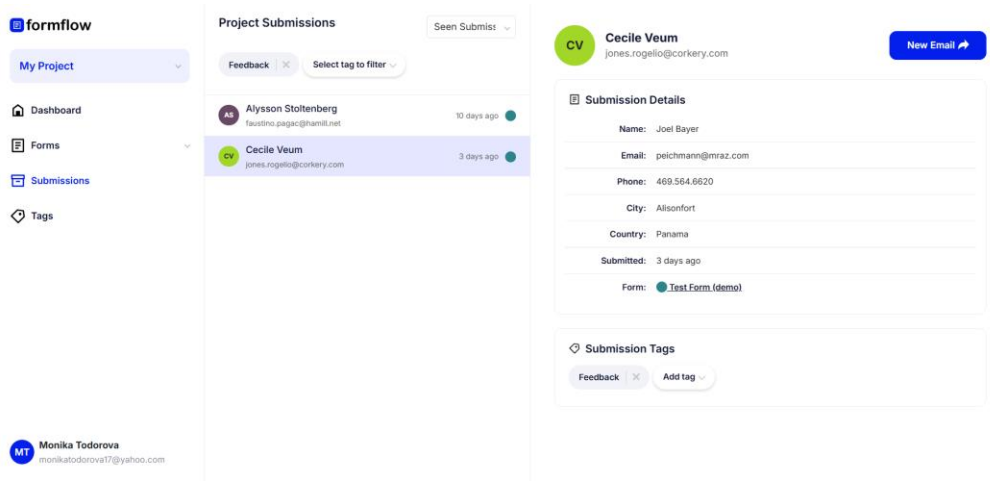
Сите одговори се соодветно означени со бојата на нивната форма, со цел полесно распознавање. Дополнително во зависност од статусот на одговорот, односно дали истиот е виден, позадината на истиот е во соодветна боја.

Со селектирање на некој од одговорите, од десна страна се покажуваат деталите поврзани со него, можност за одговор на истиот што пренасочува со mailto линк и опција за додавање на тагови.



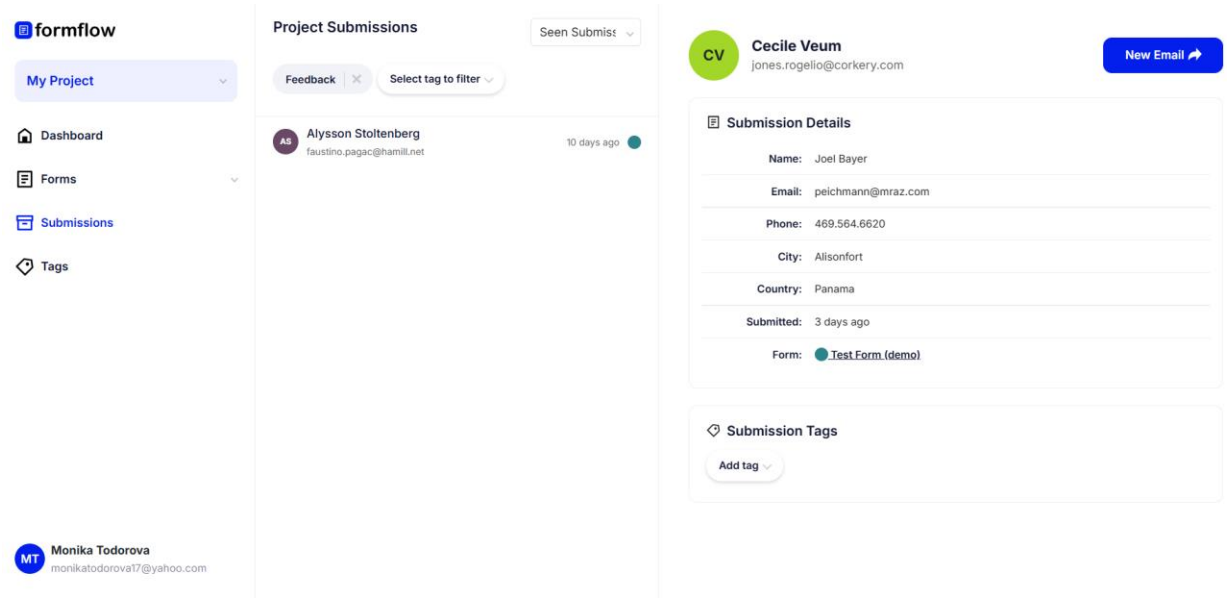
Слика 40 – Приказ на делати за одговор

Доколку некој од одговорите го поврземе со тагови, овозможено ни е филтрирање по истите. Односно за првите два одговори од сликата погоре ставаме таг Feedback, што понатаму ни овозможува филтрирање. Филтрирањето дополнително функционира и според статус, односно може да се прикажуваат само новите одговори, само прочитаните одговори или сите заедно.



Слика 41 – Филтрирање со таг

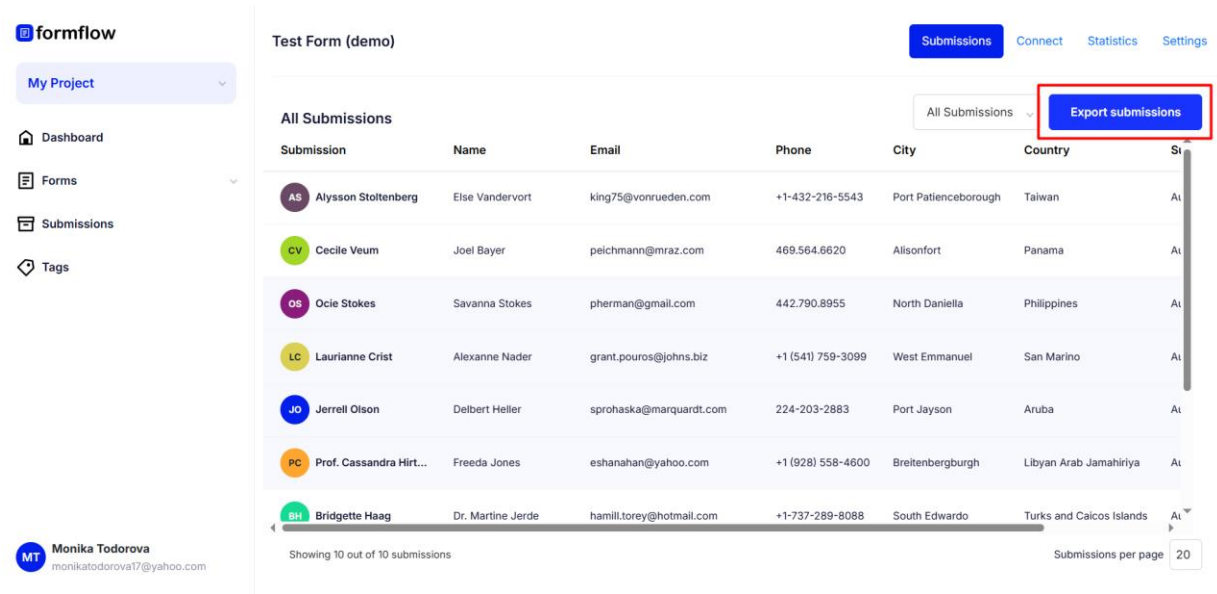
Доколку отстраниме некој од таговите во моменталниот приказ, листата автоматски се обновува.



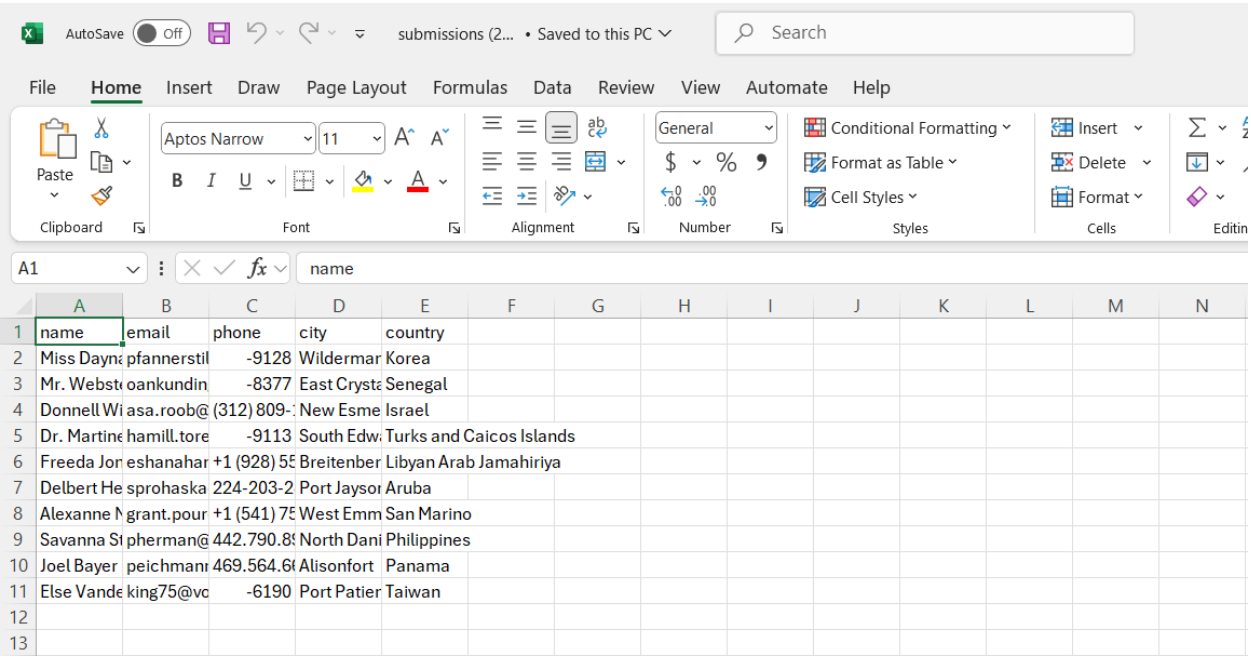
Слика 42 – Управување со таговите при филтрирање

Генерирање и преземање на документ

При приказот на сите одговори од дадена форма, овозможена е акција за презмање на одговорите во документ во csv формат.



Слика 43 – Поле за преземање на документ



Слика 44 – Приказ на преземаниот документ

Креирање и менаџирање на тагови

Кога станува збор за таговите, нивното менаџирање се одвива во посебна страна, каде може да се додаваат и бришат тагови на ниво на корисник. Улогата на таговите не е ништо друго освен едноставен начин за групирање и означување на добиените одговори.

Слика 45 – Екран за додавање и менаџирање на тагови

Заклучок

Оваа апликација ќе овозможи на сите лица кои имаат потреба од прибирање на одговори и иницинирање на онлајн комуникација преку онлајн форми, истото да го направат на едноставен и интуитивен начин.

Изработката на апликацијата со користење на технологии како Laravel, Vue, MySQL ги опфаќа сите изучени единици според предметите Имплементација на системи со слободен и отворен код и Напреден веб дизајн. Покриени се сите основни концепти за работа со наведените технологии, и притоа е креирана целосно функционална апликација.