



Универзитет „Св. Кирил и Методиј“ во Скопје
**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И
КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

ДИПЛОМСКА РАБОТА

Изработка на веб апликација за менаџирање на форми

Ментор
доц. д-р Бобан Јоксимоски

Изработила
Моника Тодорова 201082

Скопје, Декември 2024

Содржина

Апстракт	3
1. Вовед.....	4
2. Користени Технологии.....	5
2.1. Back-end технологии	5
2.2. Front-end технологии	5
2.3. База на податоци	6
3. Архитектура на апликацијата.....	6
3.1. MVC	7
3.3. MVVM	7
3.3. ER дијаграм	7
4. Имплементација	8
4.1. Back-end Laravel	8
4.2. Front-end Vue JS	15
5. Преглед на апликацијата	23
5.1. Креирање на профил и најава.....	23
5.2. Креирање и менаџирање на проекти.....	25
5.3. Креирање и менаџирање на форми.....	26
5.4. Прибирање и менаџирање на одговори	34
5.5. Приказ на одговори и менаџирање со тагови и статуси.....	36
5.6. Генерирање и преземање на документ.....	38
5.7. Креирање и менаџирање на тагови.....	39
6. Заклучок.....	40
7. Референци.....	41

Апстракт

Онлајн формите се суштински дел од секојдневната комуникација и размена на податоци во дигиталниот свет, а многу корисници се соочуваат со предизвици при нивното креирање и управување. Целта на оваа дипломска работа е да се прикаже процесот на развој на веб апликација која овозможува лесно креирање на форми и ефикасно менаџирање на добиените одговори. Преку оваа апликација, корисниците ќе можат да генерираат сопствени форми, да ги споделат за прибирање податоци, како и да ги организираат и филтрираат добиените одговори според свои потреби. Во оваа дипломска работа детално се разгледуваат архитектурата и имплементацијата на решението, функционалностите на апликацијата, како и потенцијалните подобрувања.

Клучни зборови: Laravel, Vue JS, Веб апликација, Веб форми

1. Вовед

Во современото дигитално време, онлајн формите играат важна улога во секојдневната комуникација. Тие се незаменливи алатки за прибирање податоци, организирање анкети, спроведување истражувања и обезбедување поддршка за клиентите. Сепак, креирањето и менаџирањето на овие форми може да биде сложен процес, особено за луѓе кои немаат доволно техничко познавање. Со цел да се надмине овој предизвик, е креирана апликацијата FormFlow, која претставува иновативно решение за полесно управување со онлајн веб форми и нивните одговори.

FormFlow е веб апликација дизајнирана да понуди едноставен и интуитивен начин за креирање, конфигурирање и споделување на формите. Со оглед на тоа што главен фокус на апликацијата е лесната употреба, корисниците можат лесно да ги конфигурираат своите форми без потреба од сложено програмирање. Апликацијата овозможува три опции за креирање на форми според преференца на корисникот, со сопствен код, преку form builder или со користење на вештачка интелигенција. Овозможено е и собирање и организирање на добиените одговори за соодветните форми, како и нивно категоризирање преку додавање тагови, филтрирање според статус и групирање во проекти. Преку овие функционалности, FormFlow станува моќна алатка за ефикасно управување со форми, достапна како за технички, така и за нетехнички корисници.

Системот е изграден на основа на технологии како Laravel за backend, Vue.js за frontend и MySQL за базата на податоци, што обезбедува стабилност, флексибилност и сигурност на податоците.

Овој труд ќе ги разгледа сите аспекти на развојот на FormFlow, од почетната идеја, преку дизајнот и имплементацијата, до функционалностите кои ја издвојуваат како уникатна апликација, соодветна за широк спектар на корисници.

2. Користени технологии

Апликацијата е базирана на Laravel (PHP) како рамка за back-end, MySQL база на податоци, Vue JS како рамка за front-end и Bootstrap за стилизирање и дизајн.

2.1 Backend технологии

Backend е делот од една веб-апликација кој работи во позадина и е одговорен за обработка на податоци, логиката 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 одговор во кој се содржат информации кои се прикажуваат на корисникот. Прилично сите апликации со кои се среќаваме во нашето секојдневие комуницираат со backend слој во позадина.

2.1.1 Laravel

За изработка на backend делот во FormFlow е искористен Laravel, популарен PHP фрејмворк кој претставува чест избор токму 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 и управување со бази на податоци. Дополнително, Laravel има огромна заедница и обилна документација, што овозможува лесно да се најдат решенија за различни проблеми.^[1]

2.2 Frontend технологии

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, прифаќање на кориснички внес и праќање барања до backend делот соодветно на корисничкиот внес.

2.2.1 Vue JS

Vue.js 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 овозможува креирање на реискористиви компоненти, што го прави развојот на апликацијата многу поорганизиран и лесен за одржување. Овој пристап ги разделува различните функционалности на независни делови, што ја олеснува надоградбата и проширувањето на апликацијата.

Vue.js овозможува автоматско ажурирање на корисничкиот интерфејс во реално време, што е идеално за апликации кои бараат интуитивен и динамичен интерфејс.^[2]

2.3 База на податоци

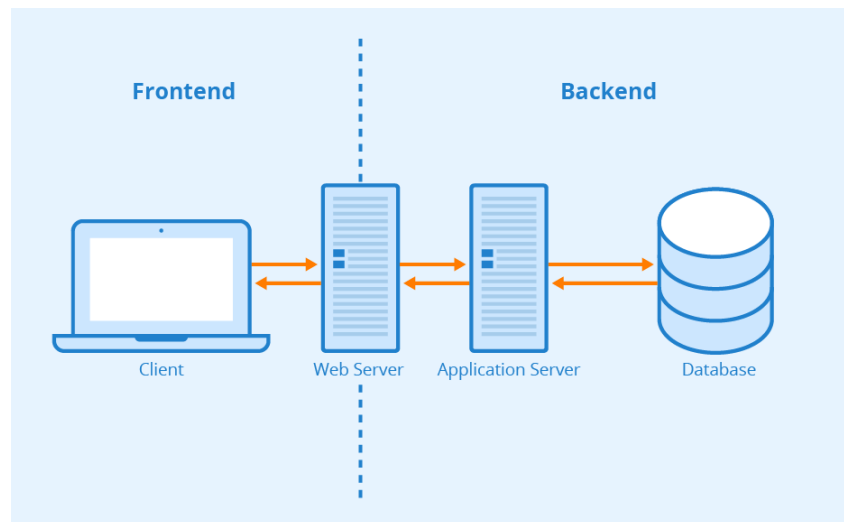
База на податоци е збирка или колекција на податоци кои опишуваат одреден елемент. Со цел комуникација со базите на податоци, земање и анализирање на податоци, постојат системи за управување со бази на податоци (DBMSs). Овозможуваат создавање, дефинирање, ажурирање, пребарување и управување со базите.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

2.3.1 MySQL

MySQL е една од релационите бази на податоци на пазарот. Нуди високи перформанси, дури и при големи количини податоци. Користи релациски модел, кој овозможува комплексни релации помеѓу различни табели. За FormFlow, ова значи дека може лесно да се организираат и управуваат податоците за различни форми и проекти. Како open-source база на податоци, MySQL е бесплатен и може да се прилагоди на потребите на апликацијата што претставува економски поволно решение за развој на апликации.

MySQL исто така има голема заедница и достапна документација, што овозможува лесно да се решат евентуални проблеми и да се добијат ресурси за оптимизација. ^[3]

3. Архитектура на апликацијата



Слика 1. Архитектура на апликацијата

Како што веќе споменавме, системот за управување со форми и одговори е изграден со архитектура која ги интегрира способностите на Vue.js за frontend, Laravel за backend и MySQL за базата на податоци.

Бидејќи апликацијата се состои всушност од два одделни дела за backend и frontend, тие имаат и засебна архитектура соодветно.

3.1 MVC

Backend делот на апликацијата изграден со Laravel ги следи принципите на MVC (Model-View-Controller) архитектура. MVC архитектурата се состои од три главни компоненти:

- **Model:** Моделите претставуваат податочни ентитети кои во комбинација со Eloquent ORM го управуваат пристапот на податоците до базата. Во нив се дефинираат и чуваат атрибутите на ентитетите од кои се состои апликацијата како и релациите помеѓу истите.
- **View:** Погледите го претставуваат корисничкиот интерфејс преку кој корисникот комуницира со апликацијата. Во нашиов случај, погледите не се искористени бидејќи frontend делот претставува засебна целина изолирана од backend делот, која комуницира преку API за размена на податоците.
- **Controller:** Контролерите ја играат главната улога на управување со податоците и одговорите кои ги добива frontendот односно корисникот. Ги обработуваат влезните барања и враќаат соодветни одговори.

Дополнително, со цел подобра организација и функционалност на кодот, воведени се уште неколку компоненти кои се вметнуваат помеѓу MVC слоевите, како сервиси и репозиториуми, за чија имплементација и улога ќе споменеме во понатамошниот текст.

3.2 MVVM

Vue JS се базира на MVVM архитектура (Model-View-ViewModel). Но, овде потребно е да се спомене дека станува збор за поразлични Model и View елементи во однос на оние од backend архитектурата.

- **Модел:** Моделите во MVVM се податоците кои се чуваат во Vue компонентата.
- **View:** Овој дел ги користи податоците од моделот за да ги прикаже на екран, и најчесто се наоѓа во *<template>* делот од компонентата.
- **ViewModel:** Ги синхронизира промените на податоците во моделот, директно во самото View. Па, така, добиваме промени кои се видливи за корисникот во реално време.

3.3 EP дијаграм

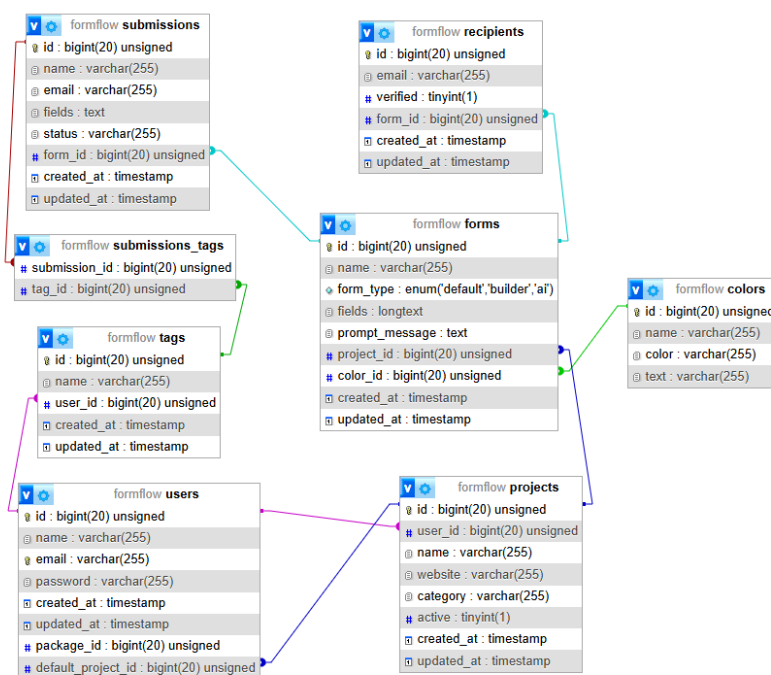
EP дијаграмот се состои од главните ентитети кои вушност ја претставуваат апликацијата и релациите помеѓу нив.

Во нашата апликација имаме неколку ентитети, меѓу кои: project, form, submission, tag, user. За сите од нив чуваме атрибут id со цел уникатна идентификација на рекордите. За сите корисници чуваме податоци за име и презиме, емаил и лозинка.

Еден корисник може да има минимум еден или повеќе проекти со цел групирање на формите, а притоа еден проект може да содржи ниту една или повеќе форми. Од овде произлегува дека еден корисник може да има ниту една или повеќе форми.

Секој од проектите и секоја од формите припаѓаат исклучиво на корисникот од кој се креирани истите.

За секоја форма може да се добиени ниту еден или повеќе различни одговори. Секој од овие одговори, може да биде означен со таг доделен од апликацијата или креиран од корисникот.



Слика 2. EP дијаграм

4. Имплементација

4.1 Backend – Laravel

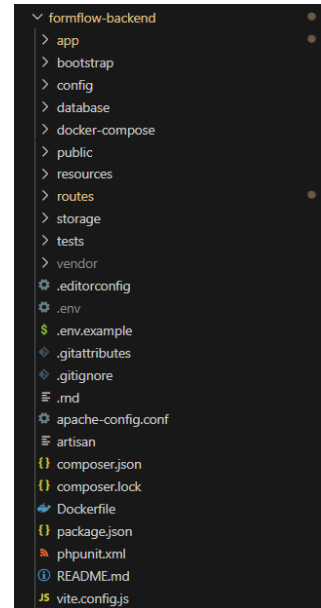
За да се креира нов Laravel проект се користи командата `laravel new formflow-backend` или ако се користи Composer `composer create-project laravel/laravel formflow-backend`.

Со ова, ќе се креира нова Laravel апликација во директориумот `formflow-backend`, која со извршување на командата `php artisan serve` ќе биде достапна за прегледување во пребарувач.

4.1.1 Структура на код

Откако ќе се креира проектот, структурата на проектот се состои од повеќе документи кои се организирани во фолдери на следниов 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.

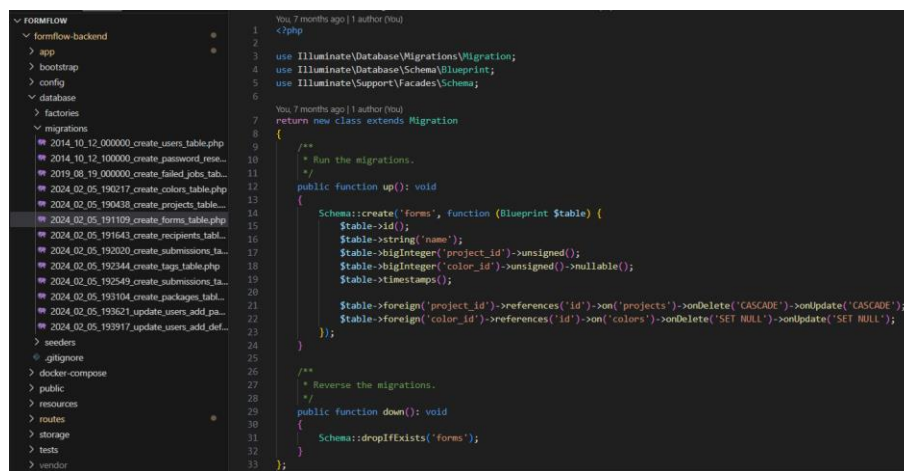


Слика 3. Структура на Laravel

4.1.2 Работа со база на податоци 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`. [4]

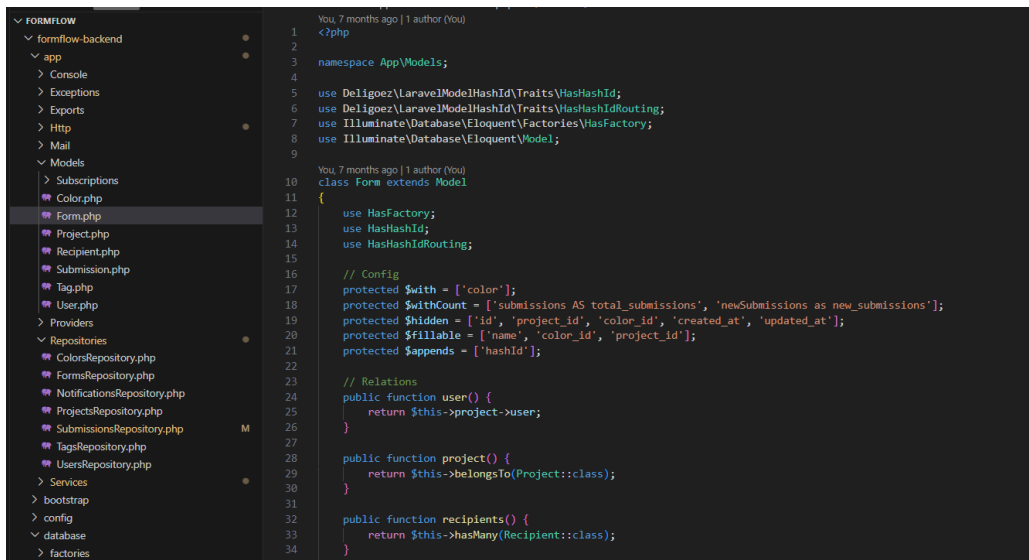


Слика 4. Пример миграција за креирање на табелата за форми

4.1.2.1 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.

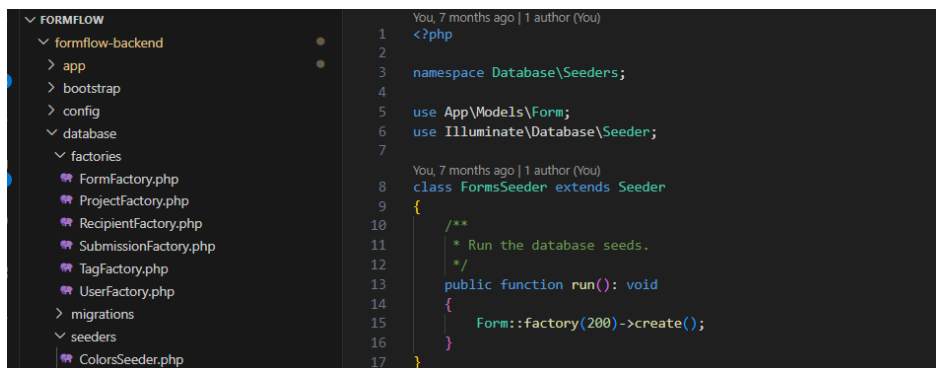


Слика 5. Модел за ентитетот форма, приказ на неколку релации

4.1.2.2 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.

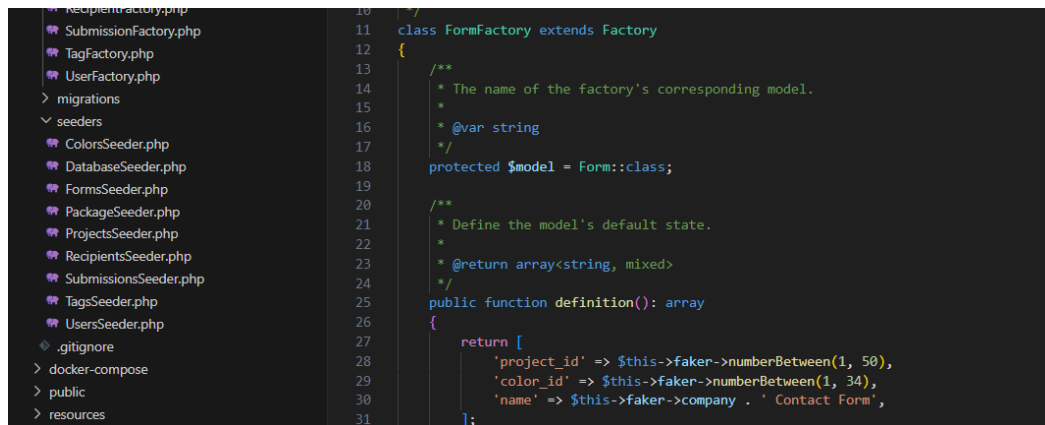
Нова seeder класа во фолдерот 'database/seeders' креираме, и ги извршуваме истите преку командите `php artisan make:seeder FormsTableSeeder` и `php artisan db:seed`



Слика 6. Seeder за ентитетот Form

4.1.2.3 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`

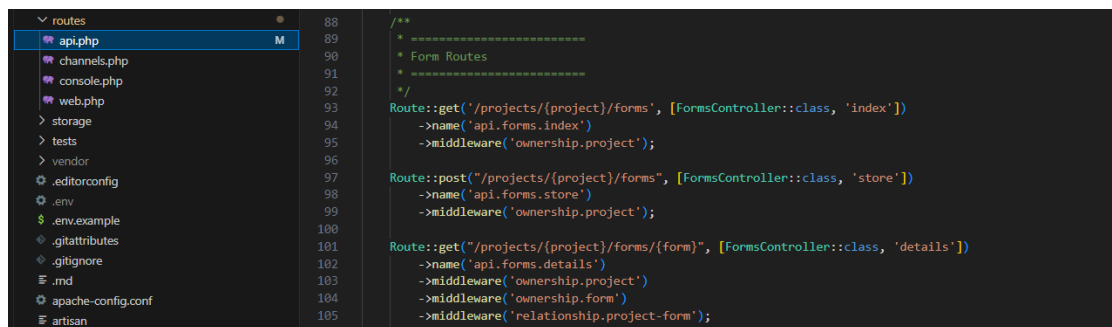


Слика 7. Factory класа за моделот Form

4.1.3 Рутирање во Laravel

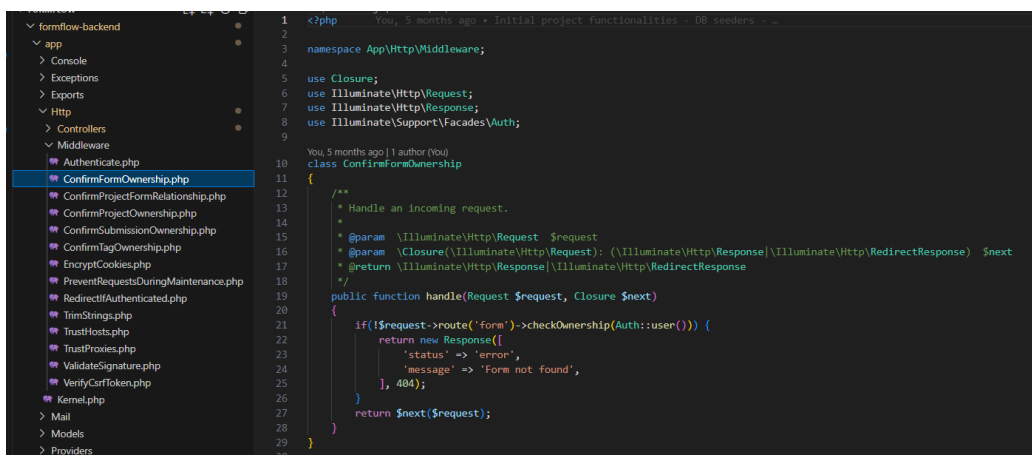
За да може frontend делот од апликацијата да комуницира со backend и базата на податоци, потребно е да се креираат API ендпоинти.

Во Laravel, овие ендпоинти, односно рутите се дефинираат во 'routes' директориумот. 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).



Тие се користат за да извршат различни проверки или операции, како што се автентикација, авторизација, валидација на податоци, проверка на кориснички дозволи и релации.

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 9. Пример middleware за проверка на сопственост на форма

4.1.4 Структура на проект

4.1.4.1 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 10. Модел за Form

4.1.4.2 Контролери

Контролерите се класи кои ги обработуваат HTTP барањата и ги враќаат одговорите. Дефинираните рути ги пренасочуваат барањата кои пристигаат со соодветна акција во одреден контролер која понатаму го обработува барањето. Тие обично се наоѓаат во `app/Http/Controllers`.

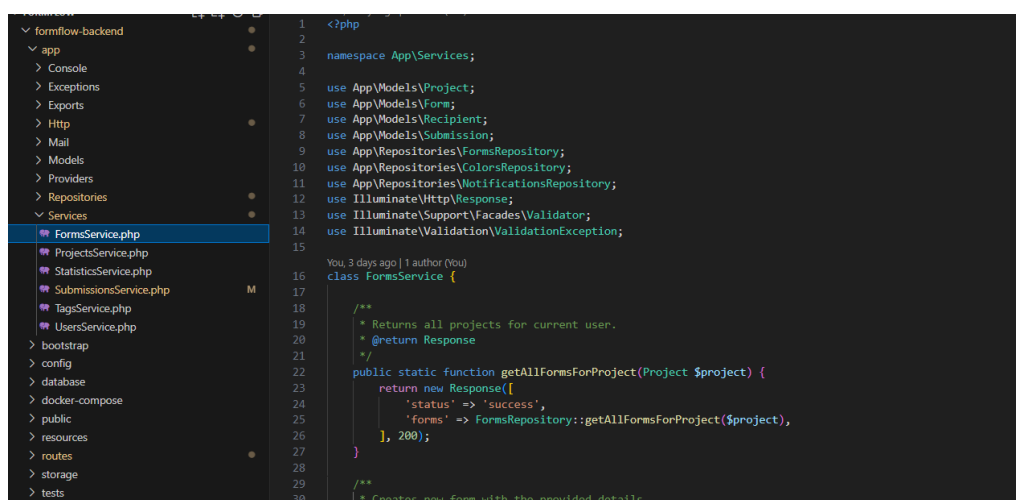


Слика 11. Контролер за Form

4.1.4.3 Сервиси

Сервисите се класи кои содржат бизнис логика која е одделена од контролерите за да се подобри организацијата и одржувањето на кодот.

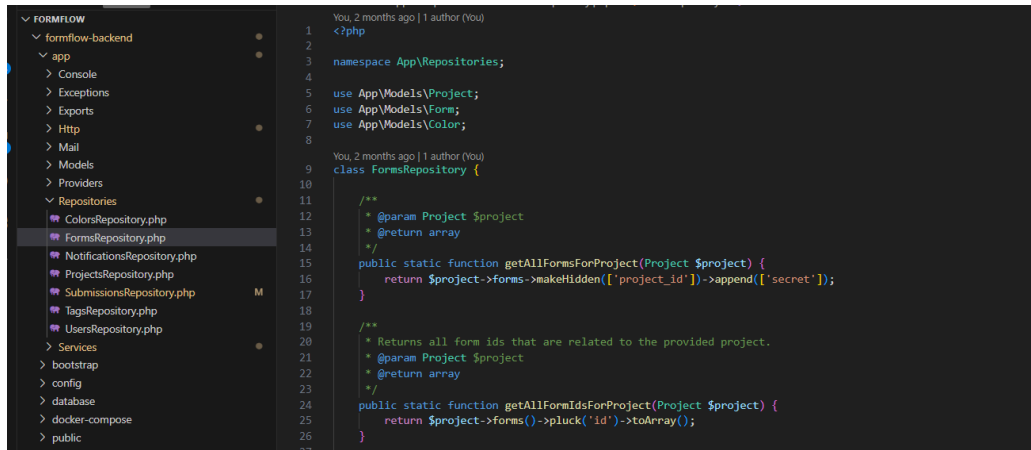
Кога ќе пристигне барање до контролерот, од таму се повикува соодветна функција од сервисот која дополнително го обработува барањето и враќа соодветен очекуван одговор до контролерот. Тие се наоѓаат во `app/Services`.



Слика 12. Сервис за Form

4.1.4.4 Репозиториуми

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



Слика 13. Репозиториум за Form

4.1.4 Обработка на API барање во Laravel

Кога ќе се испрати барање до Laravel апликацијата, изолирани во слоеви според нивната намена, се случуваат повеќе чекори:

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 функција 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 повикува и 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 комуницираат со базата преку нив за да ги добие или зачува потребните податоци.
5. **Одговорот се враќа:** Контролерот го враќа одговорот во соодветен формат (на пример, JSON).

4.1.4 Интеграција со OpenAI

Како што веќе споменавме, во рамки на апликацијата корисниците ќе можат да креираат форма која ќе биде генерирана со помош на вештачка интелигенција. Односно, корисникот

е потребно да состави порака во која ќе опише повеќе детали за формата која му е потребна. Пораката креирана од корисникот, заедно со други дополнителни детали и насоки за генерирање на соодветен одговор, се испраќаат до OpenAI. Од таму се добива одговор кој понатаму се обработува и враќа до корисникот соодветно.

Со цел испраќање на едно такво барање користејќи библиотека `Http::headers` го подготвуваме `Http` барањето кое ги содржи API клучот кој овозможува комуникација со OpenAI, како и генерираната порака од наша страна.

```
$response = Http::withHeaders([
    'Authorization' => 'Bearer ' . env('OPENAI_API_KEY'),
])->post('https://api.openai.com/v1/chat/completions', [
    'model' => 'gpt-4',
    'messages' => [
        ['role' => 'system', 'content' => 'You are a form generator.'],
        ['role' => 'user', 'content' => $prompt],
    ],
]);
$generatedForm = $response->json('choices.0.message.content');
$formFields = json_decode($generatedForm, true);
```

Слика 14. Интеграција со вештачка интелигенција

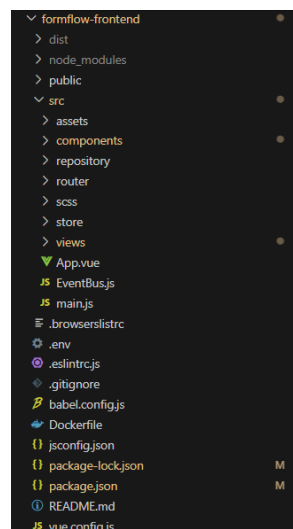
4.2 Frontend – Vue JS

За да се креира нов Vue.js 3 проект, може да се користи Vue CLI или Vite, притоа се извршуваат командите: `npm create vite@latest my-vue-app -- --template vue`

4.2.1 Структура на кодот

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 проектот:

- **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



Слика 15. Структура во Vue

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`.

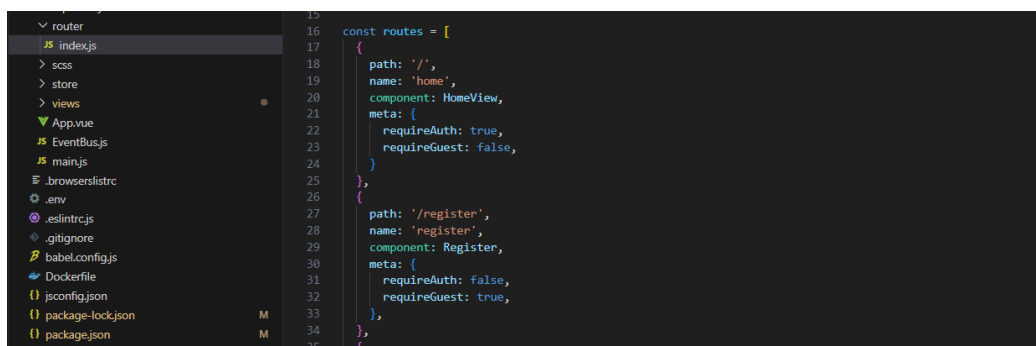
4.2.2 Рутирање во Vue

Рутирањето е клучен концепт во развојот на апликации од една страна (SPA) со Vue.js. 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.

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.

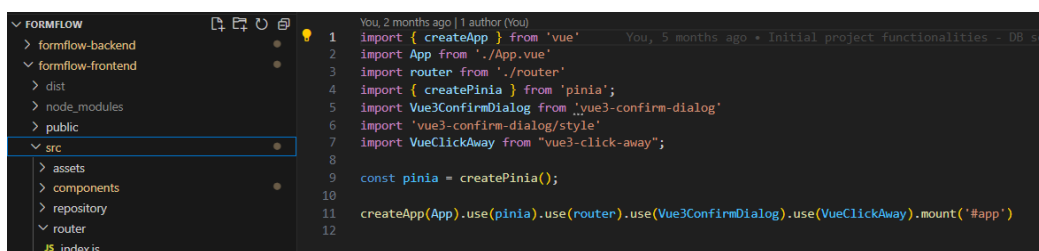
Во Vue.js 3, рутирањето се имплементира со помош на **Vue Router**, кој се инсталира со помош на командата `npm install vue-router`

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 16. 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 17. Интеграција на vue-router


```

24
25     <li class="nav-item">
26       <router-link to="/submissions" class="nav-link" exact>
27         
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         
35         <span>Tags</span>
36       </router-link>
37     </li>

```

Слика 18. Употреба на vue router

4.2.3 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 сме сигурни 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.

4.2.3.1 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.

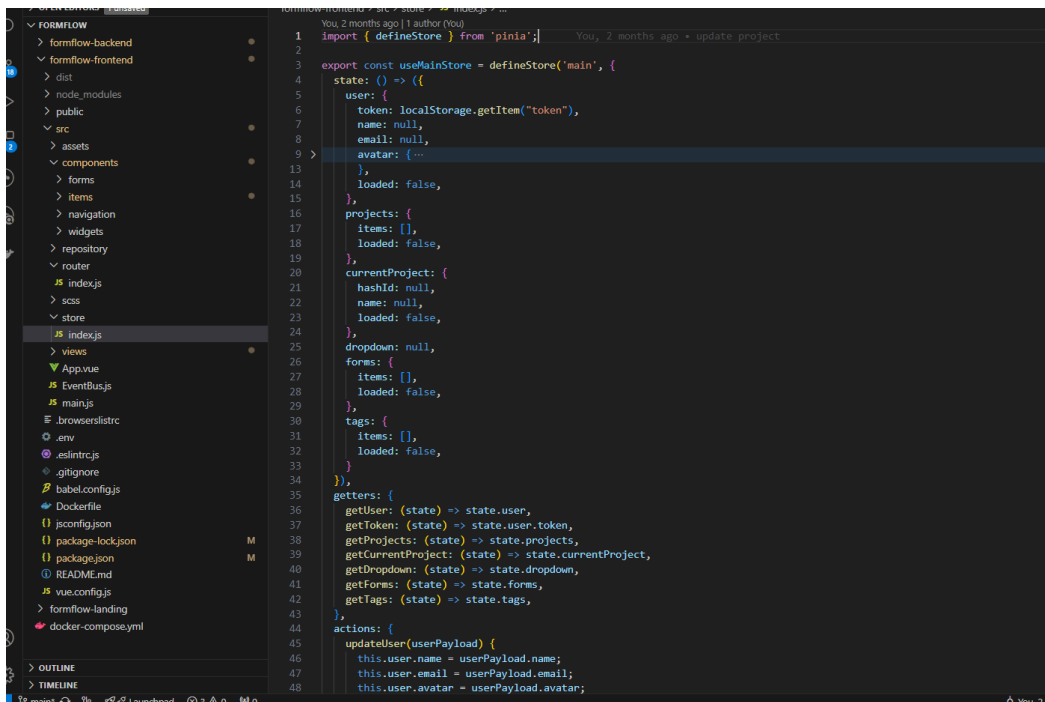
За инсталирање на Pinia се користи командата `npm install pinia`.

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

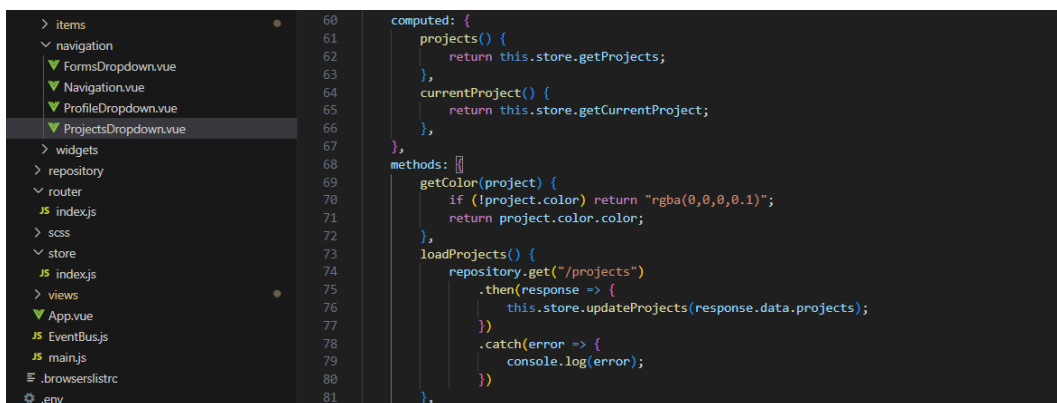
По интеграцијата се креира Store со Pinia, каде што се дефинираат:

- **state:** Објект кој ги содржи сите податоци кои ќе бидат користени во Store-от.
- **getters:** Функции кои овозможуваат да се пристапат податоците
- **actions:** Методите кои овозможуваат да се менува состојбата

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



Слика 19. Store во Vue



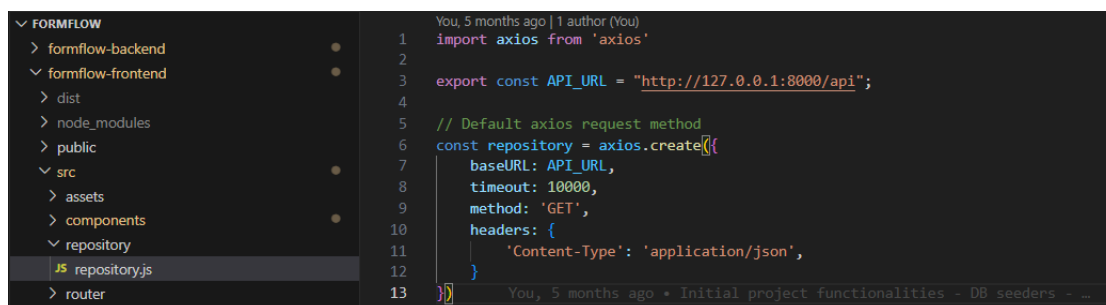
Слика 20. Употреба на store

4.2.4 Axios

Axios е популарна JavaScript библиотека што се користи за правење HTTP барања од клиентската страна до серверската страна. Со други зборови, Axios овозможува комуникација помеѓу frontend на апликацијата и серверот, каде што се наоѓа backend.

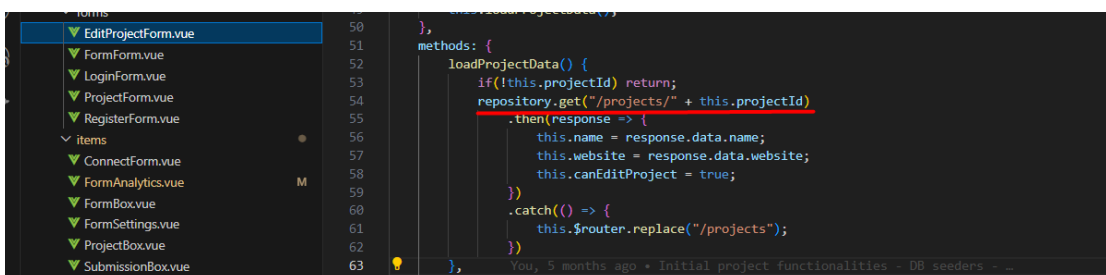
Ова е многу важно за веб-апликации, каде што корисниците внесуваат податоци и сакаат да ги испратат до серверот за обработка и чување, или пак сакаат да добијат информации од серверот за прикажување.

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, каде што се дефинираат основните параметри како `API_URL` со цел да не мора на секој повик да се внесува целиосниот url.

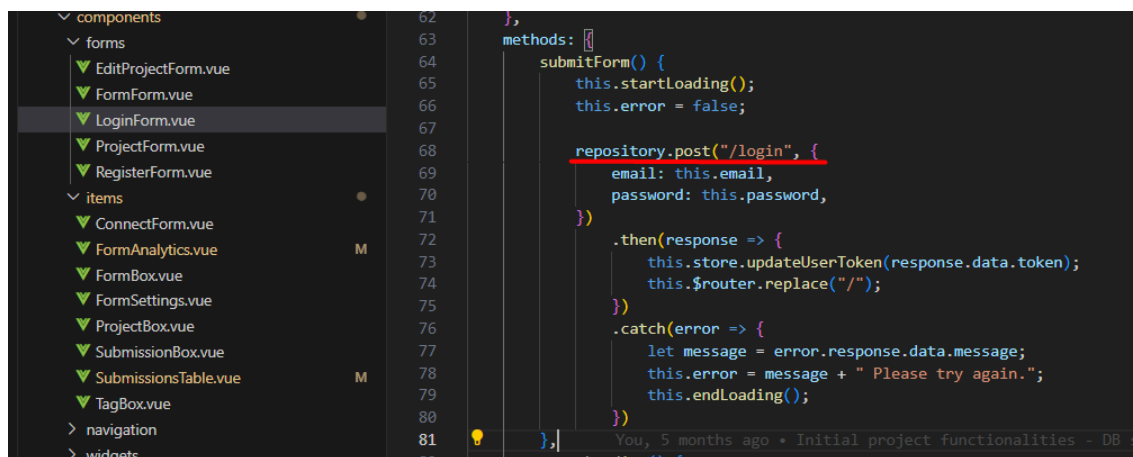


Слика 21. Конфигурација на axios

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



Слика 22. Употреба на axios за повици за читање од бекенд

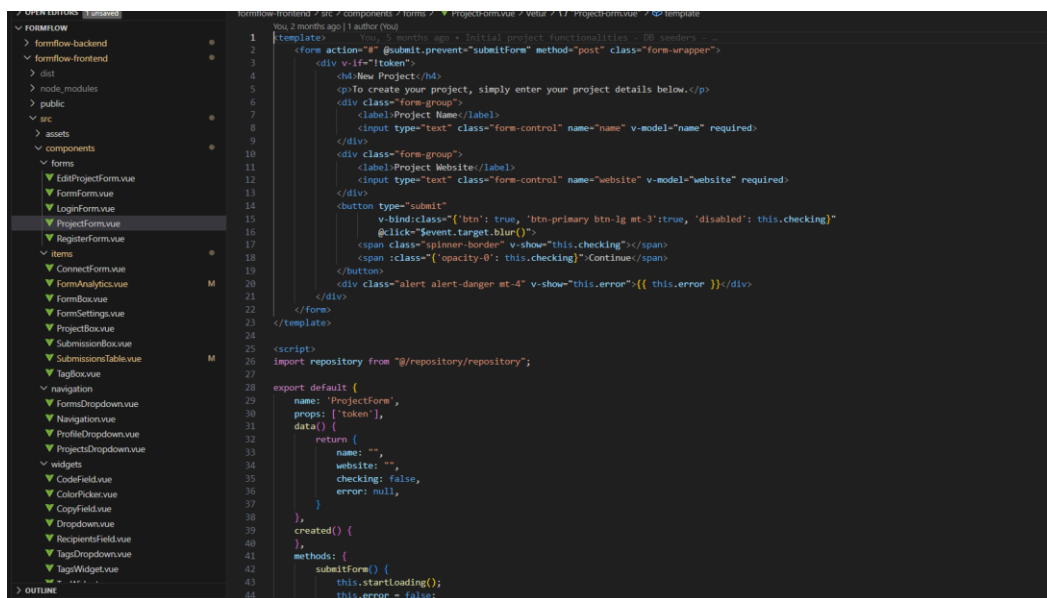


Слика 23. Употреба на axios за повици за запишување до бекенд

4.2.5 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 неопходни за правилно функционирање. [5]

Како компоненти може да се креираат најразлични елементи како форми, табели, копчиња, па дури и цели страници.



Слика 24. Пример компонента

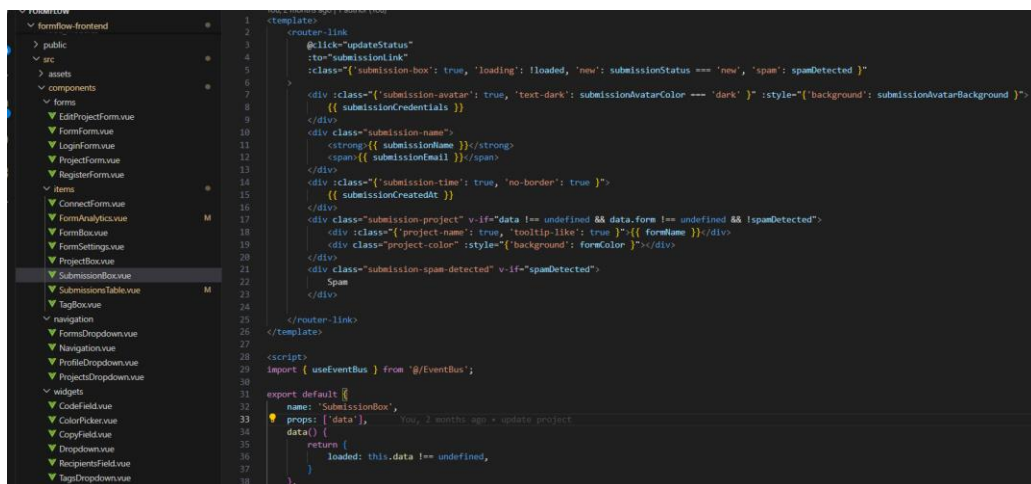
Во примеров е прикажана компонента за форма за креирање проект насловена, која понатаму е интегрирана во одредена страница и се прикажува според одредени услови – со пристап со специфична рута од апликацијата. Компонентава служи за целосно справување со процесот на додавање на нов проект во апликацијата.

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

4.2.5.1 Props

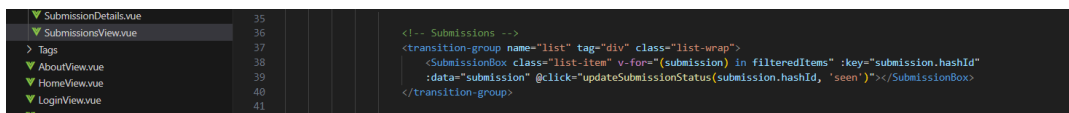
Props се кратенки за “properties” и се користат за пренесување на податоци од родител компонентата до компонентата дете. Ова овозможува компонентите да бидат пофлексибилни и да се употребуваат со различни вредности.

За комуникација пак помеѓу компоненти кои не се во релација родител – дете се користи 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 мора компонентите да се во релација за да се испраќаат податоците како props.



Слика 25. 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 од одредена форма (Form Submissions), така што за секој одговор се користи истата компонента, но со различни податоци соодветно кои се испраќаат како prop со :data="submission".



Слика 26. Испраќање на параметар како prop

4.2.5.2 Methods

Методи во Vue.js се функции кои се дефинираат во <script> 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 во нашата апликација.

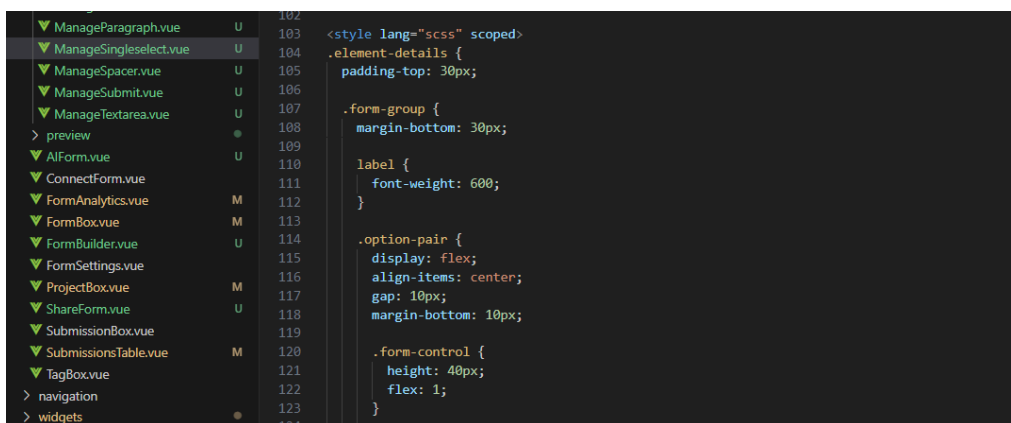


Слика 27. Приказ на дефинирани методи

Во примеров на слика 27 се прикажани неколку акции како читање и листање на таговите, додавање на одреден таг за одреден одговор и справување со клик при додавање на таг.

4.2.5.3 Стили

Во Vue може да се креираат стилови кои ќе важат глобално низ целата апликација или за одреден сегмент. Во секоја компонента може со користење на тагот <style> да се дефинираат правилата за стилизирање.

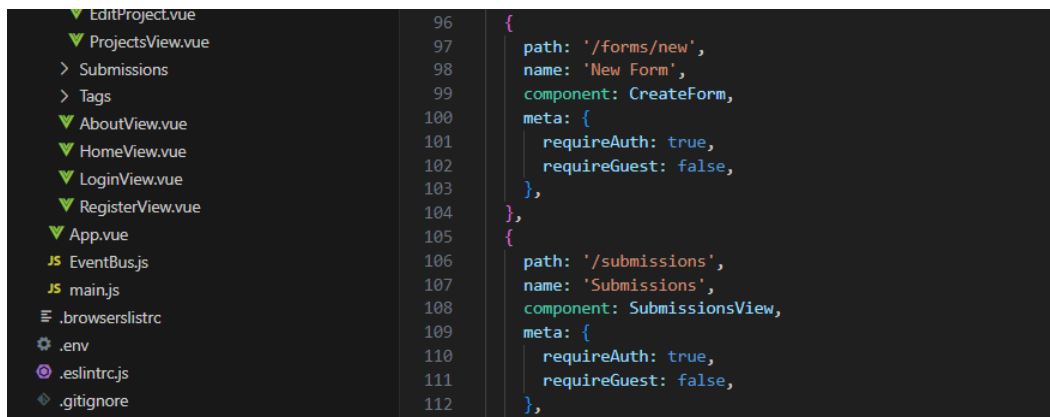


Слика 28. Приказ на дефиниција за стил во одреден сегмент

4.2.6 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 вчитуваат врз основа на рутата дефинирана во Vue Router.

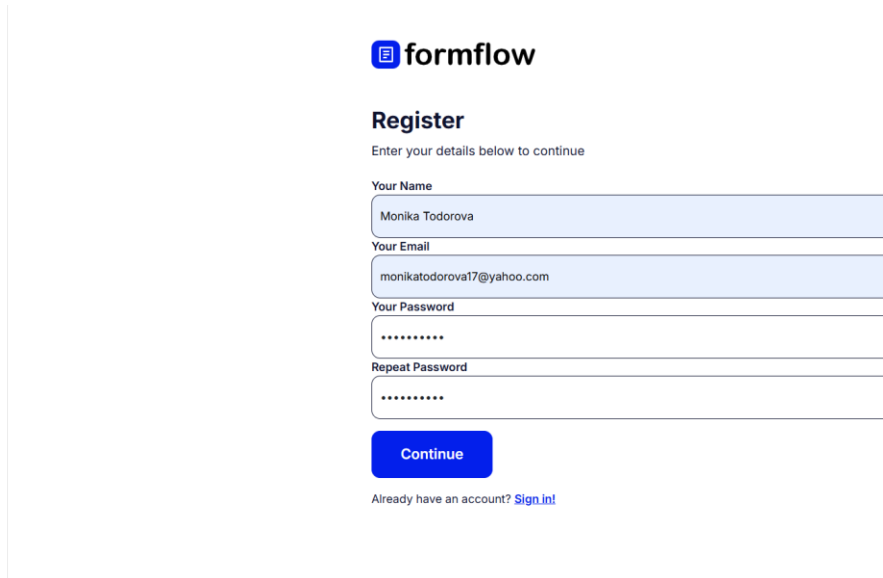


Слика 29. Страници поврзани со рути

5. Преглед на апликацијата

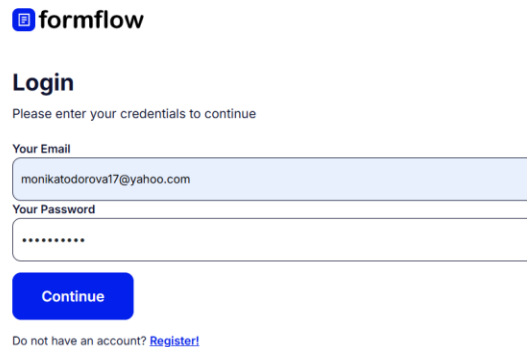
5.1 Креирање на профил и најава

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



The screenshot shows the 'formflow' logo at the top. Below it is the title 'Register' followed by the instruction 'Enter your details below to continue'. There are four input fields: 'Your Name' (containing 'Monika Todorova'), 'Your Email' (containing 'monikatodorova17@yahoo.com'), 'Your Password' (containing seven dots), and 'Repeat Password' (containing seven dots). A blue 'Continue' button is positioned below the password fields. At the bottom, there is a link: 'Already have an account? [Sign in!](#)'.

Слика 30. Екран за креирање на профил



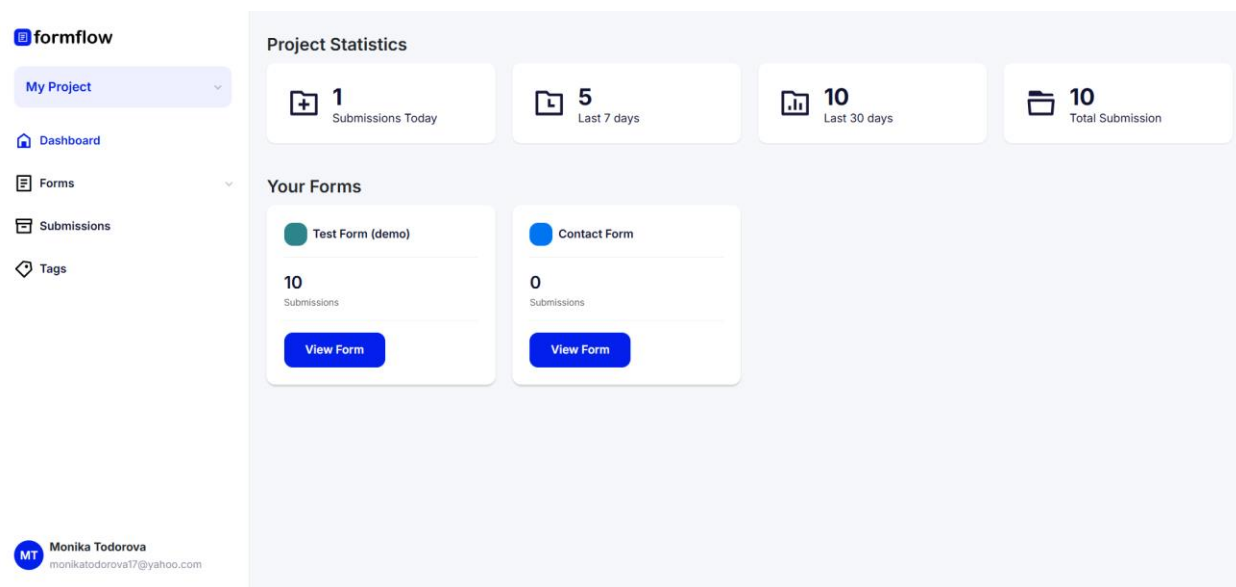
The screenshot shows the 'formflow' logo at the top. Below it is the title 'Login' followed by the instruction 'Please enter your credentials to continue'. There are two input fields: 'Your Email' (containing 'monikatodorova17@yahoo.com') and 'Your Password' (containing seven dots). A blue 'Continue' button is positioned below the password field. At the bottom, there is a link: 'Do not have an account? [Register!](#)'.

Слика 31. Екран за најава

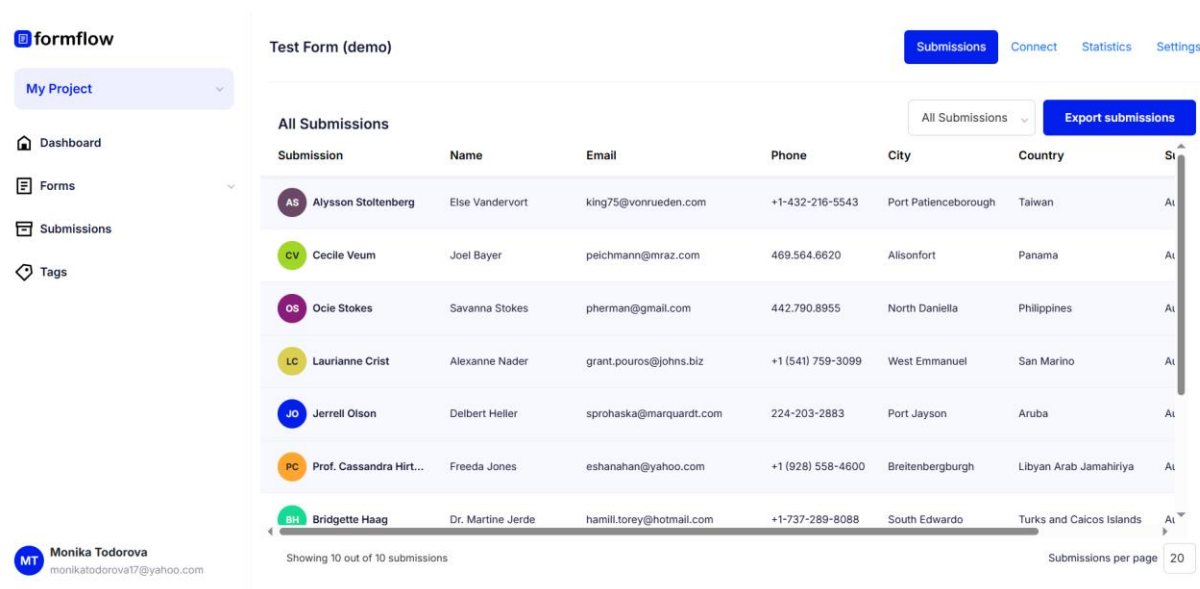
Со креирање на профилот на корисникот се креира и иницијален проект, кој се поставува како стандарден / селектиран. За истиот проект се креираат и две форми.

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 добива одговори за неа.



Слика 32. Иницијален екран - Приказ на формите на корисникот за одреден проект



Слика 33. Приказ на тест форма

5.2 Креирање и менаџирање на проекти

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

Групирање по проект е замислено да се користи 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 според своите потреби.

Креирањето на проект се одвива со внесување на бараните 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 да се 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.

The screenshot shows the 'formflow' application interface. On the left is a sidebar menu with 'My Project' selected, and options for 'Dashboard', 'Forms', 'Submissions', and 'Tags'. The main area is titled 'New Project' with the instruction 'To create your project, simply enter your project details below.' It contains two input fields: 'Project Name' with the value 'Test project' and 'Project Website' with the value 'test.com'. A blue 'Continue' button is at the bottom. The footer shows the user 'Monika Todorova' with a profile icon and email 'monikatodorova17@yahoo.com'.

Слика 34. Екран за креирање на нов проект

The screenshot shows the 'formflow' application interface with the 'Projects' section active. The sidebar menu on the left has 'My Project' selected, and the 'Your projects' section shows 'Test project' and 'My Project' (selected). Below this are buttons for 'Create new project' and 'Manage projects'. The main area is titled 'Projects' and displays two project cards. The first card is for 'Test project' (test.com) with 0 Forms and 0 Submissions, and buttons for 'Edit' and 'Delete'. The second card is for 'My Project' (No website) with 2 Forms and 10 Submissions, and buttons for 'Edit' and 'Delete'. The footer shows the user 'Monika Todorova' with a profile icon and email 'monikatodorova17@yahoo.com'.

Слика 35. Екран за приказ на креираните проекти

5.3 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 за форми во навигацијата.

За да се креира една форма, потребно е само да се внесе нејзино име, а може да се селектира и некоја од понудените бои, со единствена цел полесно распознавање на формата и одговорите поврзани со неа.

formflow

My Project

Dashboard

Forms

Your forms

- Test Form (demo)
- Contact Form

Create new form

New Form

Enter the following details to create your form. The color you choose is used for better representation of the forms and their analytics.

Form Name

Test Form

Color

Save form

Monika Todorova
monikatodorova17@yahoo.com

Слика 36. Екран за креирање на нова форма

formflow

My Project

Dashboard

Forms

Submissions

Tags

Your form has been created

Complete the process by building your form using the form builder or AI, or continue with your own code

Form Builder

Build and customize your forms without any coding knowledge required.

Custom Code

Start accepting submissions from your custom coded form instantly.

AI Form

Enter a prompt to specify the form fields and get the form created by AI.

Monika Todorova
monikatodorova17@yahoo.com

Слика 37. Екран за избор на тип на форма – form builder

По креирањето на формата, автоматски корисникот се пренасочува на екран за избор на тип на форма. Во зависност од изборот кој ќе го направи корисникот, ќе добие соодветни опции за креирање и споделување на формата. Понудени се три можности, односно form builder, изворен код и со користење на вештачка интелигенција.

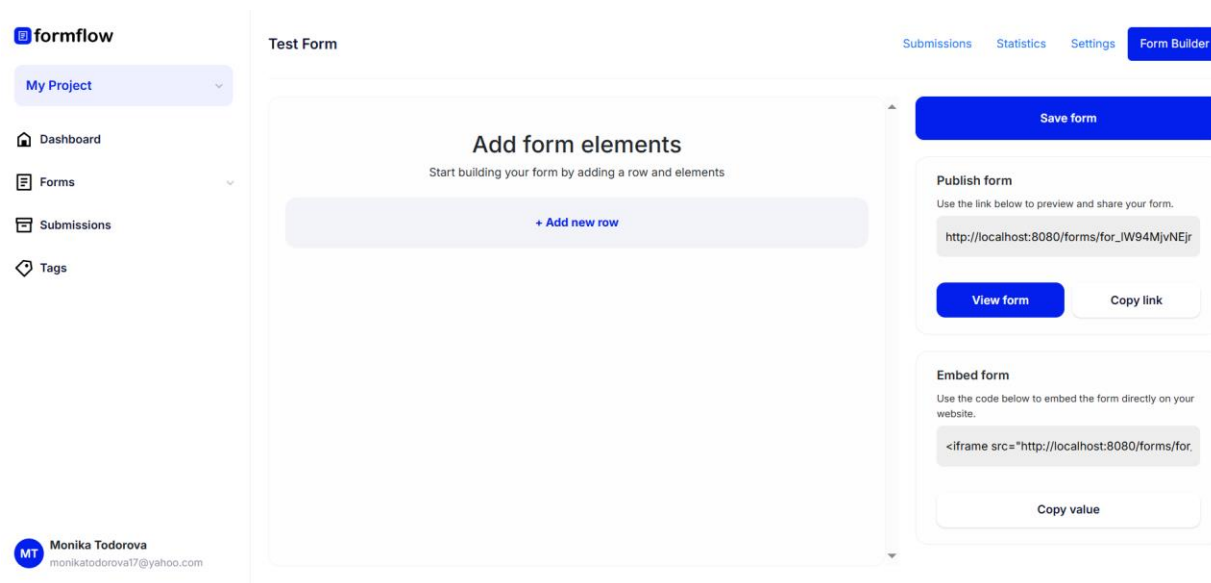
Доколку корисникот не селектира ниту една од опциите, понатаму кога ќе сака да пристапи до формата, повторно ќе му се појави овој екран. Односно, не може да се пристапи до екраните за менаџирање на формата се додека не се дефинира нејзиниот тип.

Во секој случај, новокреираната форма ќе биде прикажана на екранот за приказ на форми од одреден проект, како на Слика 31.

5.3.1 Form builder

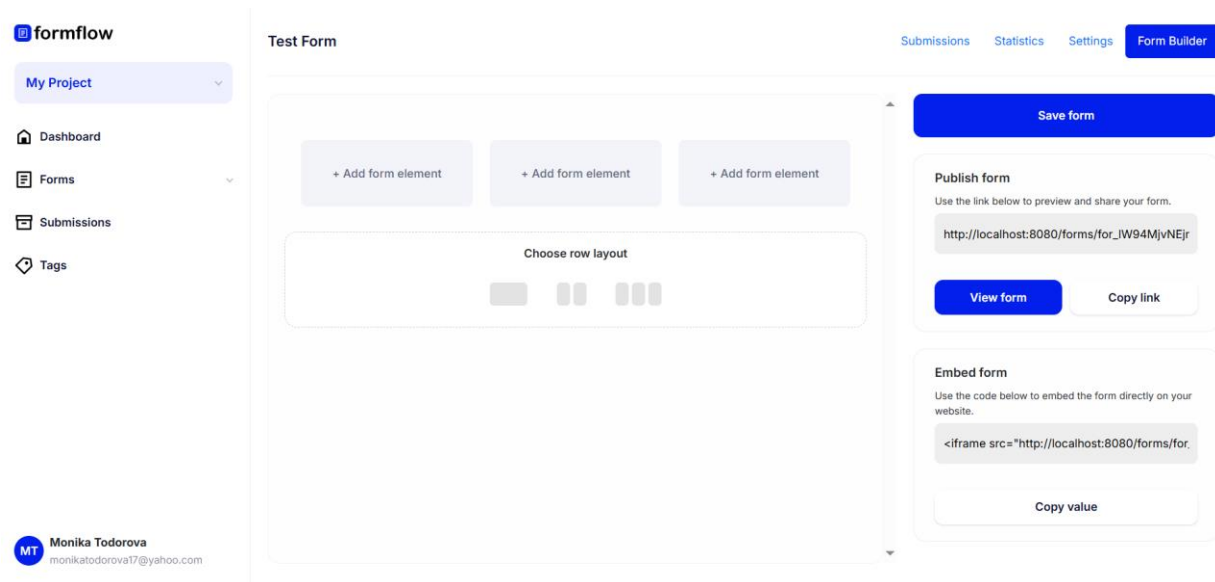
Како прва опција за тип на форма е прикажана опцијата form builder. Доколку корисникот ја селектира оваа опција, ќе добие можност да ја изгради својата форма користејќи ги предефинираните елементи, и притоа ќе може истите да ги прилагоди според своите потреби.

По избирањето на овој тип на форма, корисникот се пренасочува на екран каде може да ја види својата моментална форма. Со оглед на тоа дека формата е празна, не се прикажани елементи, туку само опцијата за додавање на нов ред во архитектурата.



Слика 38. Екран при избор на тип form builder

Кога ќе додаде нов ред, корисникот може да избере една од трите опции за број на колони во редот, прикажано на Слика 39. Имено, во еден ред елементите може да бидат распоредени во една, две или три колони соодветно на изборот на корисникот.

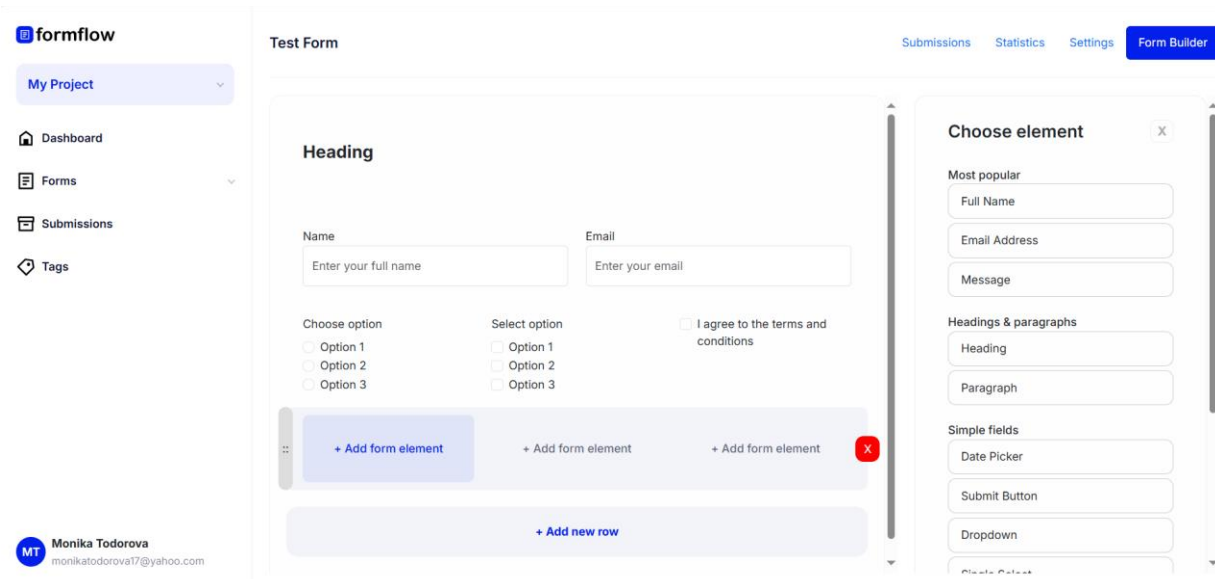


Слика 39. Екран за градење на форма

Откако корисникот ќе додаде нов ред во одреден распоред, со клик на Add form element полето, му се прикажува страничен елемент каде се излистани сите понудени полиња за форма, подредени според категорија.

Во опциите овде се прикажани прилично сите најкористени типови на полиња како, инпут полиња, полиња за текст, избор на датум, избор на една или повеќе опции, листа за селекција, како и полиња за стилизирање на формата, односно наслови, параграфи, линии за сепарација или елементи за празни места.

Со клик на некоја од опциите, истата се поставува во селектираното поле во формата.



Слика 40. Екран за додавања на полиња

За новододаденото поле да може да се менаџира, се клика на полето во формата и повторно се отвора страничниот елемент, но со поразлична содржина. Формата за менаџирање на полето се разликува за сите елементи, во зависност од типот на елементот и кои опции е потребно да може да се менаџираат од страна на корисникот.

За обичното поле за внес, овозможено е менаџирање на лабелата, текстот во полето, како и дали полето во формата е задолжително или не. Односно дали во полето мора да биде внесена вредност, за да може формата да се испрати.

The screenshot shows the 'formflow' application interface. On the left is a sidebar with 'My Project' selected, and links to 'Dashboard', 'Forms', 'Submissions', and 'Tags'. The main area is titled 'Test Form' and contains a form with fields for 'Name' and 'Email', a 'Choose option' section with three radio buttons, a 'Select option' section with three radio buttons, and a checkbox for 'I agree to the terms and conditions'. A blue 'Submit' button is at the bottom. On the right, the 'Manage Element' panel is open, showing fields for 'Label' (set to 'Name'), 'Placeholder' (set to 'Enter your full name'), and 'Required' (set to 'Yes'). There are 'Update' and 'Delete' buttons at the bottom of the panel.

Слика 41. Екран за менаџирање а полиња

Доколку пак избереме поле за избор на еден или повеќе одговори, се нуди можност за едитирање на лабелата, но и на вредностите кои се понудени како опции. Секоја од опциите може да се промени или да се избрише, а може да се додаваат и нови опции.

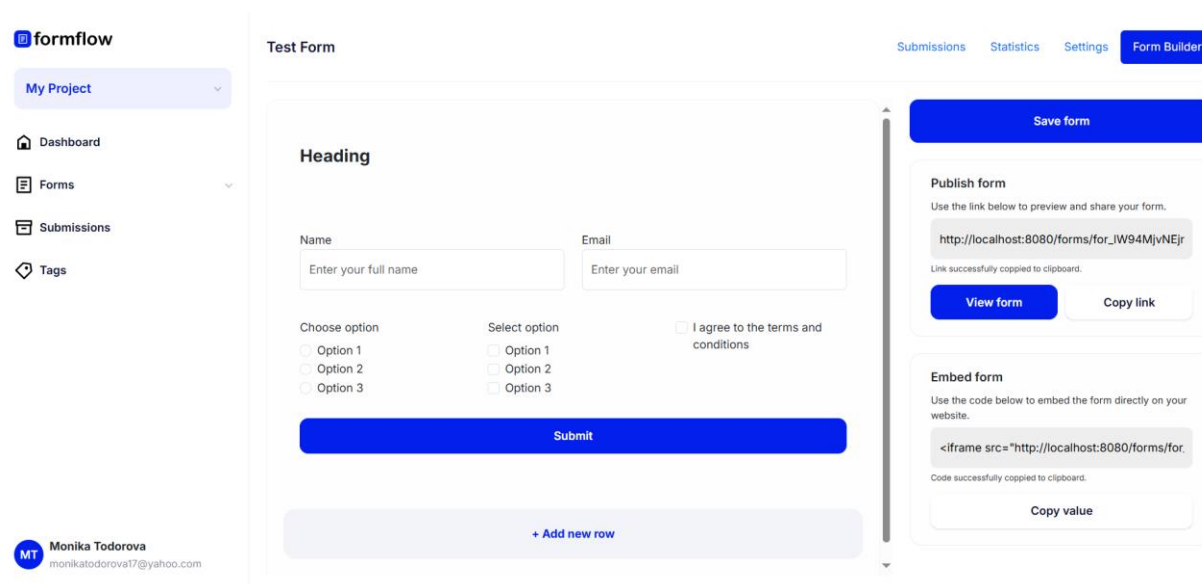
The screenshot shows the 'formflow' application interface. On the left is a sidebar with 'My Project' selected, and links to 'Dashboard', 'Forms', 'Submissions', and 'Tags'. The main area is titled 'Test Form' and contains a form with fields for 'Name' and 'Email', a 'Choose option' section with three radio buttons, a 'Select option' section with three radio buttons, and a checkbox for 'I agree to the terms and conditions'. A blue 'Submit' button is at the bottom. On the right, the 'Manage Element' panel is open, showing fields for 'Label' (set to 'Choose option') and 'Options'. There are three options listed: 'Option 1' with value 'option-0-opt', 'Option 2' with value 'option-1-opt', and 'Option 3' with value 'option-2-opt'. Each option has a red 'X' button next to it. There is a '+ Add Option' button and a 'Default Selected' field at the bottom of the panel.

Слика 42. Екран за менаџирање на полиња за избор

Откако корисникот ќе ја изгради формата според своите потреби, потребно е да кликне на полето Save form со цел промените да бидат зачувани. По зачувување на промените, корисникот може да ја сподели или ембедира својата форма.

За споделување на формата се генерира страна во апликацијата за која не е потребно корисникот да биде автентичиран за да пристапи до неа. На оваа страна се прикажува идентитчно истата форма која корисникот ја има креирано. Притоа истата може да се пополнува и испраќа, по што корисникот ќе ги добива одговорите во апликацијата. Преку овој линк, формата може да биде споделена, и ќе биде достапна за сите кои го поседуваат соодветниот линк.

Процесот за ембедирање пак се состои од код за ембедирање кој корисникот може да го ископира преку апликацијата и да го вметне како iframe код во своја страница. Исто како и претходно, формата ќе биде прикажана идентично како што корисникот ја изградил и ќе може да се пополнува и испраќа.

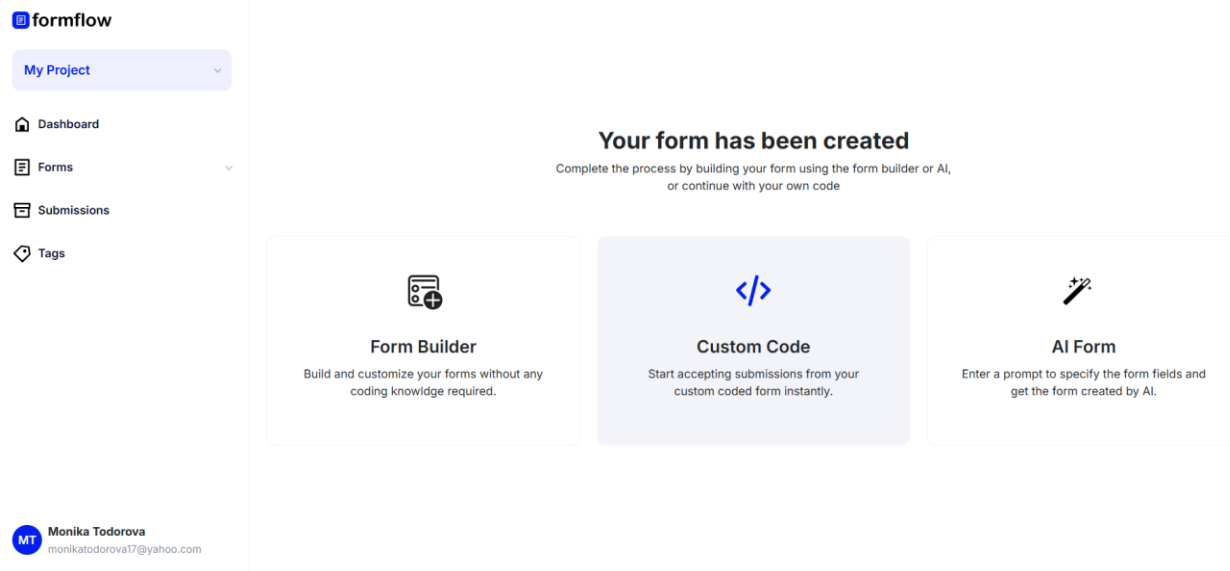


Слика 43. Екран за приказ на изградена форма

Овој начин за креирање на форма е идеален за корисници без технички познавања. Би им овозможил на многу лесен и интуитивен начин да ја креираат потребната форма, да ја споделат и потоа да ги менаџираат одговорите од истата.

5.3.2 Default code

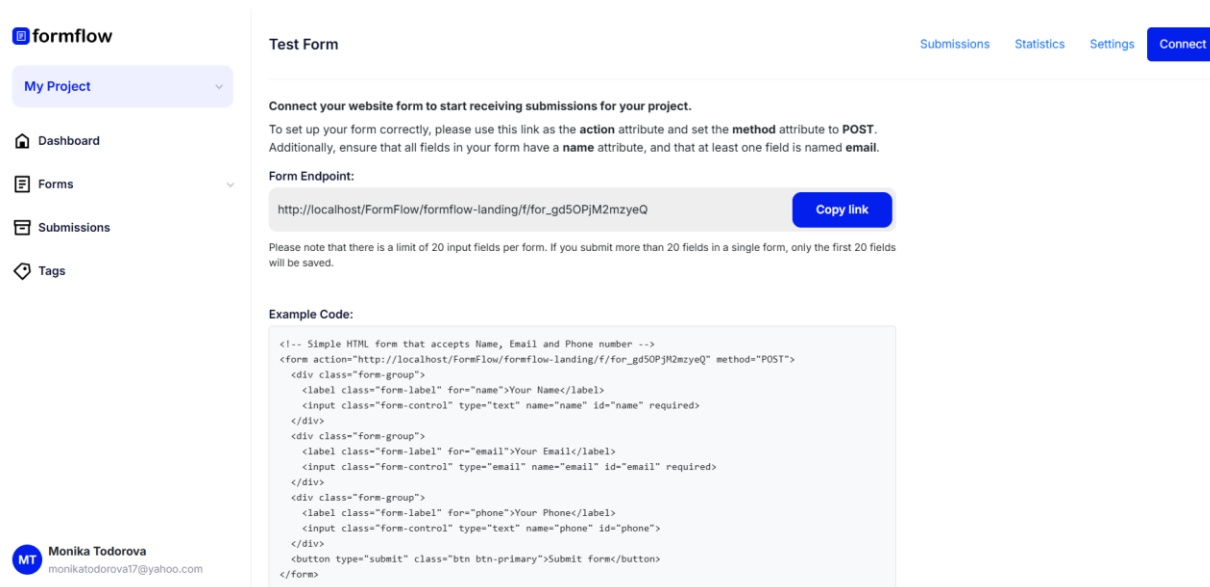
Друга понудена опција за креирање на форма е опцијата со код. При селектирање на оваа опција, корисникот нема можност визуелно да ја гради својата форма преку form builder со избор на структура и полиња. Оваа опција нуди само линк кој се поставува во action полето во form елементот, и со тоа формата се поврзува со ендпоинт во апликацијата што овозможува одговорите да пристигаат во апликацијата.



Слика 44. Екран за избор на тип на форма – custom code

Дополнително, овде се прикажан и пример код на една најобична форма, што му дава до знаење на корисникот како може да ја поврзе својата форма со апликацијата.

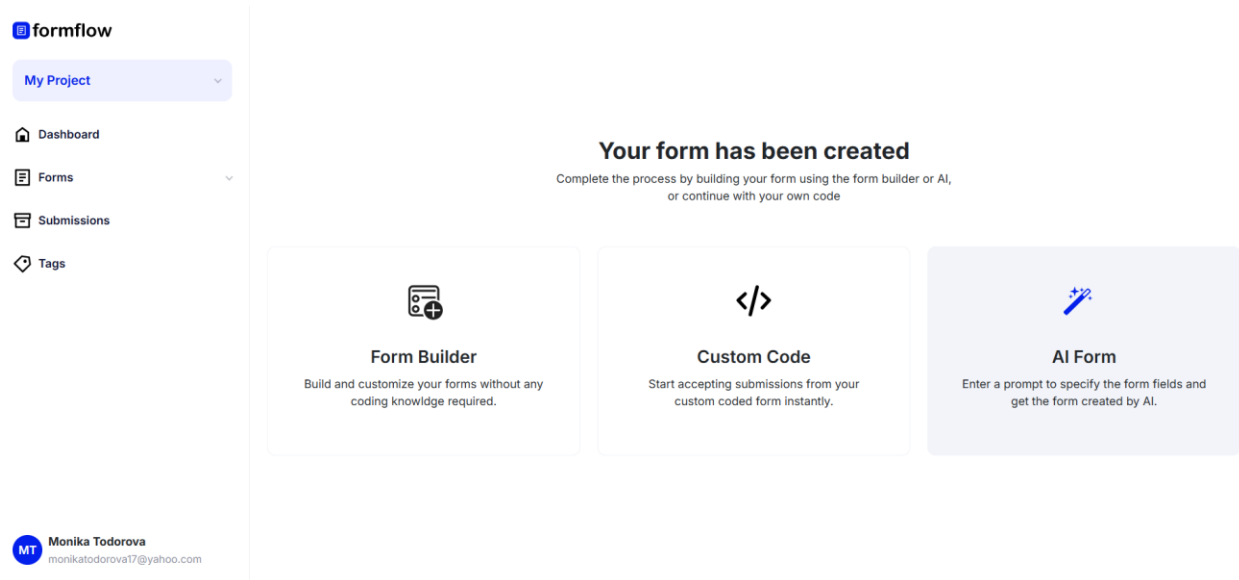
Овој начин за креирање на форми е многу пофлексибилен во однос на градењето на форми со form builder, токму од причина што корисникот има целосна слобода да го изгенерира кодот за формата како што посакува. Но, самото генерирање на кодот за формата е предизвикувачки за нетехнички лица, па овој начин за работа се препорачува повеќе за лица со поголемо техничко познавање.



Слика 45. Екран за приказ на форма со код

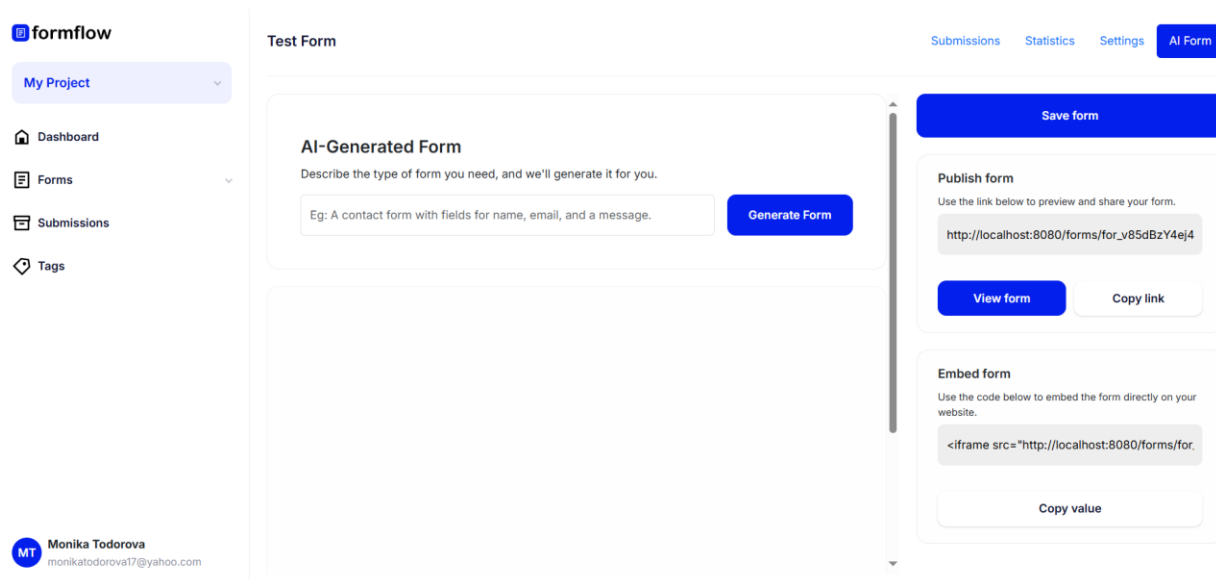
5.3.3 AI Form

Како трета и последна опција за креирање на форма е опцијата со користење на вештачка интелигенција.



Слика 46. Екран за избор на тип на форма – AI form

Оваа опција му нуди на корисникот поле за внес на опис кој би требало да ја дефинира формата која му е потребна на корисникот. По внесување на описот, со клик на полето Generate Form се испраќа барање до OpenAI, моделот gpt-4 за генерирање на соодветна форма.



Слика 47. Екран за генерирање на AI форма

Вештачката интелигенција враќа одговор во кој се содржат полињата соодветни на барањата на корисникот и истите се прикажуваат исто како кај опцијата за градење со form builder, при што корисникот повторно ги има можностите за менаџирање на секое од полињата засебно, како и можноста да додава нови полиња.

За секој форма од типот на форма со вештачка интелигенција, може да се испраќа само по едно барање до OpenAI. Односно одкако корисникот еднаш ќе испрати барање, неговиот внес ќе се прикажува над формата, и опцијата за повторно испраќање на барање ќе биде оневозможена. Доколку корисникот сака повторно да испрати барање, може да го стори истото преку креирање на нова форма од овој тип.

The screenshot shows the 'formflow' dashboard. On the left is a sidebar with 'My Project' and navigation links for 'Dashboard', 'Forms', 'Submissions', and 'Tags'. The main area is titled 'Test Form' and contains an 'AI-Generated Form' section with a prompt: "I need a contact form with fields for name email and message." Below this is a form with three fields: 'Name' (with a red 'X' icon), 'Email', and 'Message'. On the right is a 'Manage Element' sidebar with input fields for 'Label' (Name), 'Placeholder' (Enter your name), and 'Required'. It also has 'Update' and 'Delete' buttons. The bottom left corner shows the user 'Monika Todorova' with an email address.

Слика 48. Екран за менаџирање на генерираната форма

Исто како и претходно, и оваа форма може да биде споделена преку јавен линк или ембедирана во код. При што ќе се рендерира идентично како што корисникот ја креирал и зачувал, ќе може да се пополнува и испраќа.

The screenshot shows a generated form titled 'Heading'. It has two input fields: 'Name' (with placeholder 'Enter your full name') and 'Email' (with placeholder 'Enter your email'). Below these are two columns of radio button options. The first column is labeled 'Choose option' and has three options: 'Option 1', 'Option 2', and 'Option 3'. The second column is labeled 'Select option' and also has three options: 'Option 1', 'Option 2', and 'Option 3'. To the right of these is a checkbox labeled 'I agree to the terms and conditions'. At the bottom is a blue 'Submit' button.

Слика 49. Приказ на генерирана форма

Heading

Name Email

Choose option ☒ Option 1 ☐ Option 2 ☐ Option 3

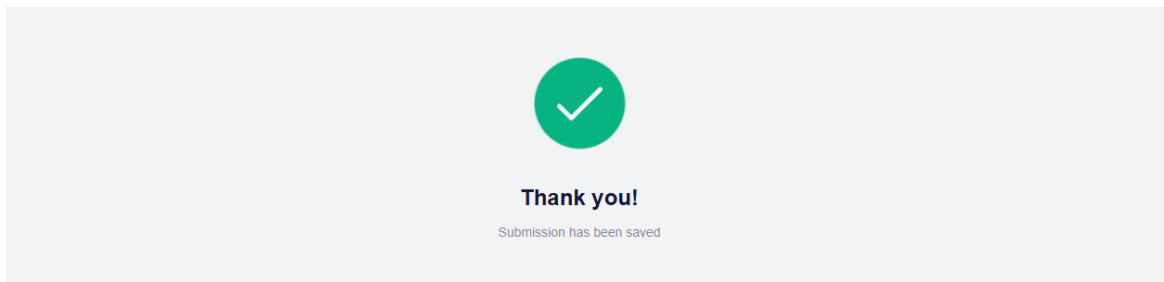
Select option ☒ Option 1 ☐ Option 2 ☐ Option 3

☒ I agree to the terms and conditions

Слика 50. Приказ на пример пополнета форма

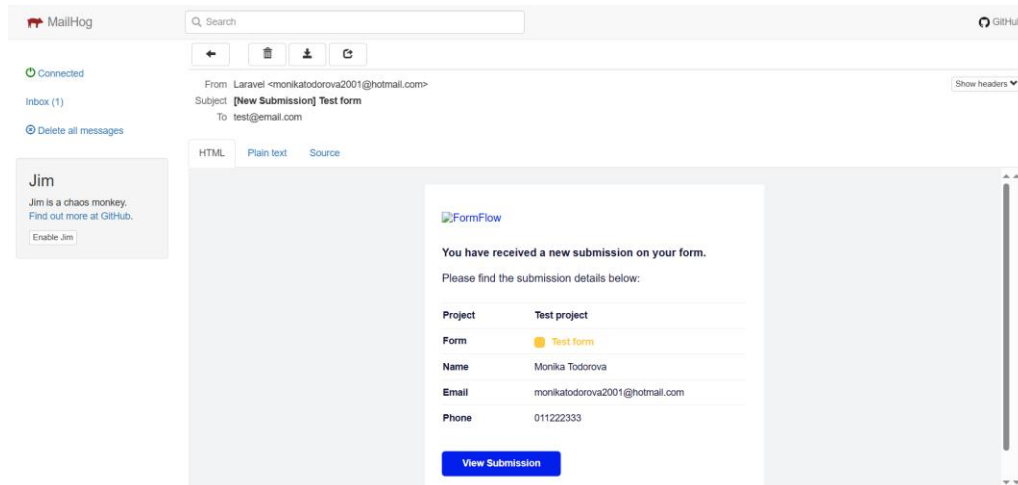
5.4 Прибирање и менаџирање на одговори

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

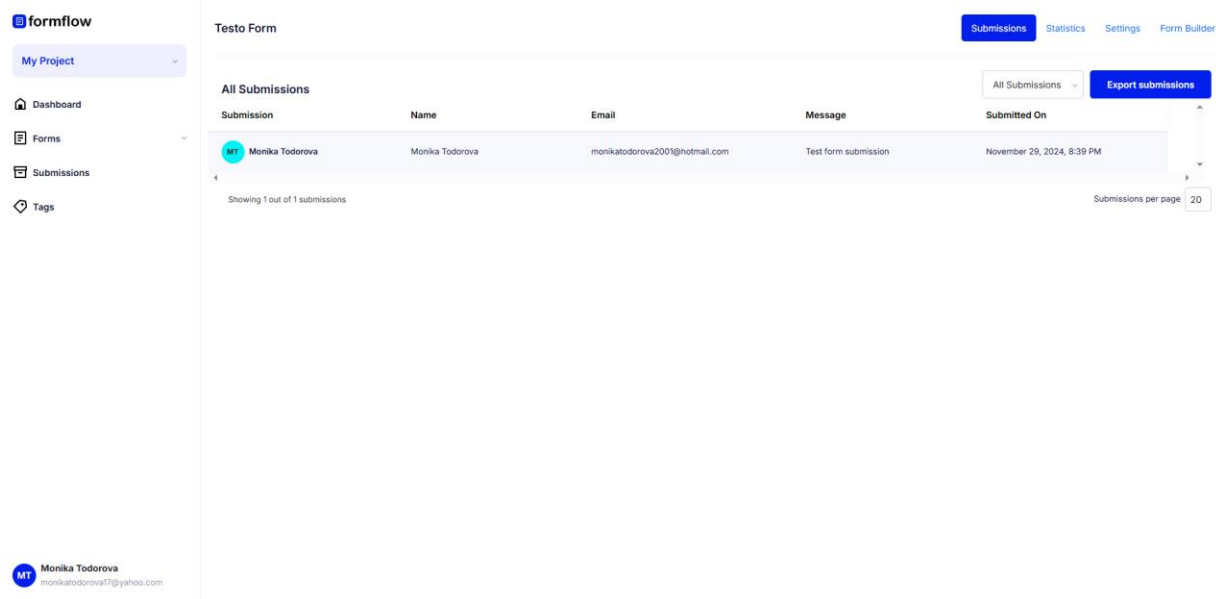


Слика 51. Успешно испраќање на одговор преку креираната форма

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.



Слика 52. Емаил известување за новодобиениот одговор



Слика 53. Приказ на новодобиениот одговор во апликацијата

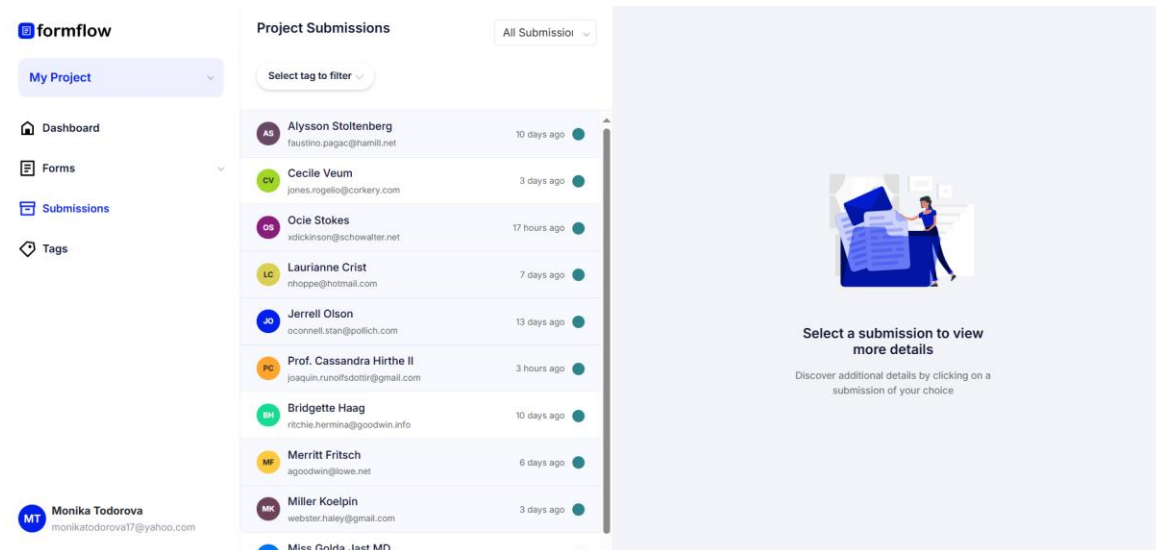
За секоја од формите е обезбеден дел со аналитика каде што може да се види по текот на претходниот месец, по ниво на денови, колкав е бројот на добиени одговори.



Слика 54. Приказ на екран со аналитики

5.5 Приказ на одговори и менаџирање со тагови и статуси

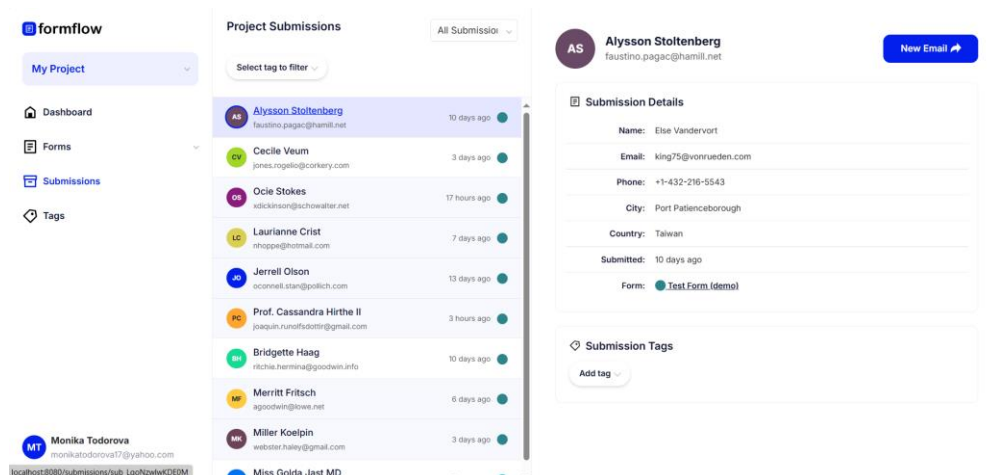
Освен приказот на одговорите поврзани со една форма, овозможен е и приказ на одговорите поврзани со целиот проект. Во делот на submissions се излистани сите одговори кои се поврзани со некоја од формите во моментално селектираниот проект.



Слика 55. Приказ на сите одговори од даден проект

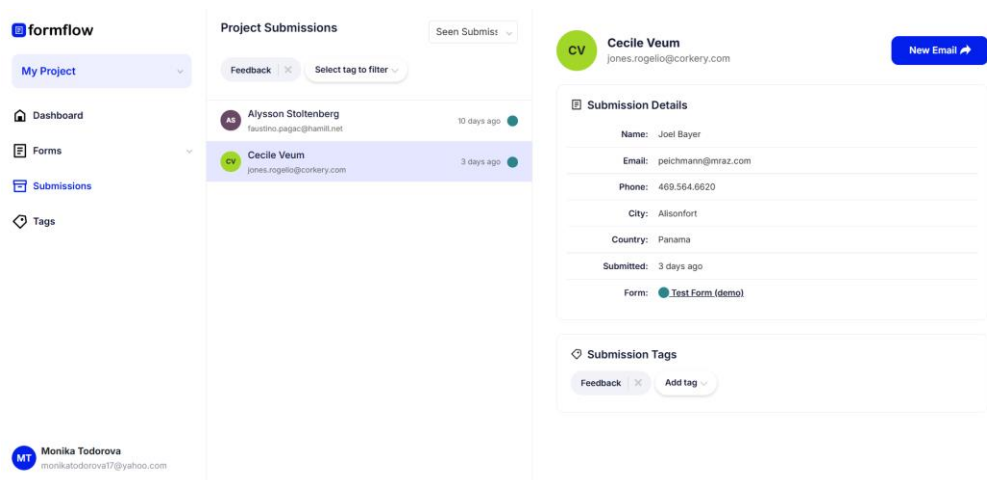
Сите одговори се соодветно означени со бојата на нивната форма, со цел полесно распознавање. Дополнително во зависност од статусот на одговорот, односно дали истиот е виџен, со цел полесна дистинкција, позадината на истиот е во соодветна поразлична боја.

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



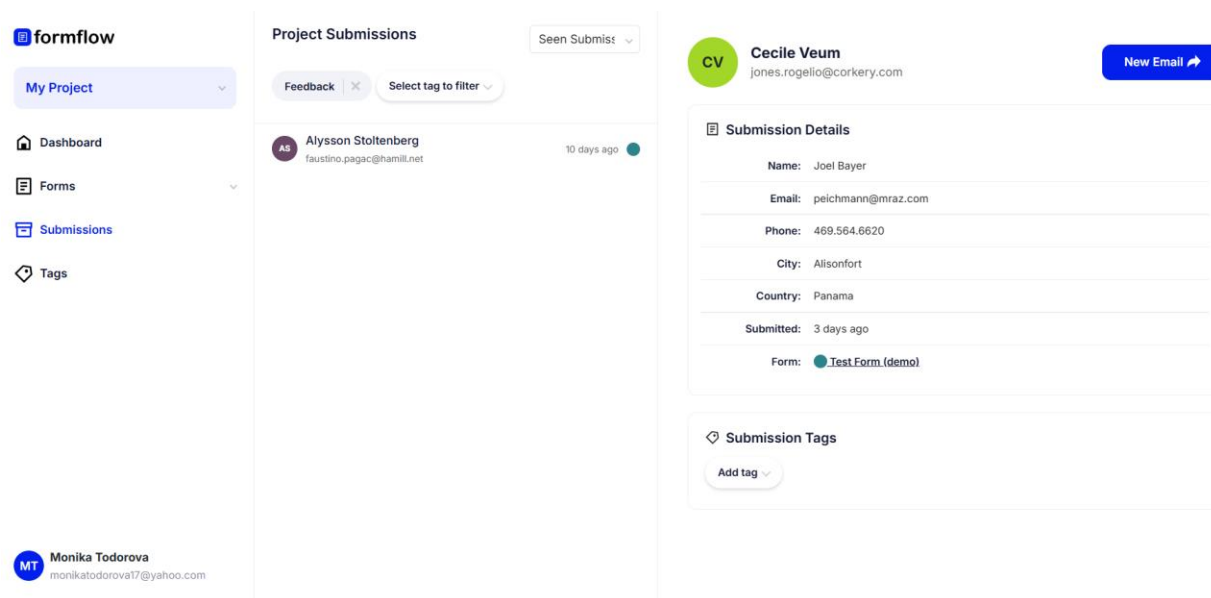
Слика 56. Приказ на делати за одговор

Доколку некој од одговорите е означен со тагови, овозможено ни е филтрирање по истите. На пример, 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 57. Филтрирање со таг

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



Слика 58. Управување со таговите при филтрирање

5.6 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.

The screenshot shows the 'formflow' interface. On the left is a sidebar with 'My Project' and navigation links for 'Dashboard', 'Forms', 'Submissions', and 'Tags'. The main area is titled 'Test Form (demo)' and contains a table of 'All Submissions'. The table has columns: Submission, Name, Email, Phone, City, Country, and Status. The 'Export submissions' button is highlighted with a red rectangle. Below the table, it says 'Showing 10 out of 10 submissions' and 'Submissions per page 20'.

Submission	Name	Email	Phone	City	Country	Status	
AS	Alysson Stollenberg	Else Vandervort	king75@vonrueden.com	+1-432-216-5543	Port Patienceborough	Taiwan	At
CV	Cecile Veum	Joel Bayer	peichmann@mrz.com	469.564.6620	Alisonfort	Panama	At
OS	Ocie Stokes	Savanna Stokes	pherman@gmail.com	442.790.8955	North Daniella	Philippines	At
LC	Laurianne Crist	Alexanne Nader	grant.pouros@johns.biz	+1 (541) 759-3099	West Emmanuel	San Marino	At
JO	Jerrell Olson	Delbert Heller	sprohaska@marquardt.com	224-203-2883	Port Jayson	Aruba	At
PC	Prof. Cassandra Hirt...	Freedra Jones	eshanahan@yahoo.com	+1 (928) 558-4600	Breitenberghurgh	Libyan Arab Jamahiriya	At
BH	Bridgette Haag	Dr. Martine Jerde	hamill.torey@hotmail.com	+1-737-289-8088	South Edwardo	Turks and Caicos Islands	At

Слика 59. Поле за преземање на документ

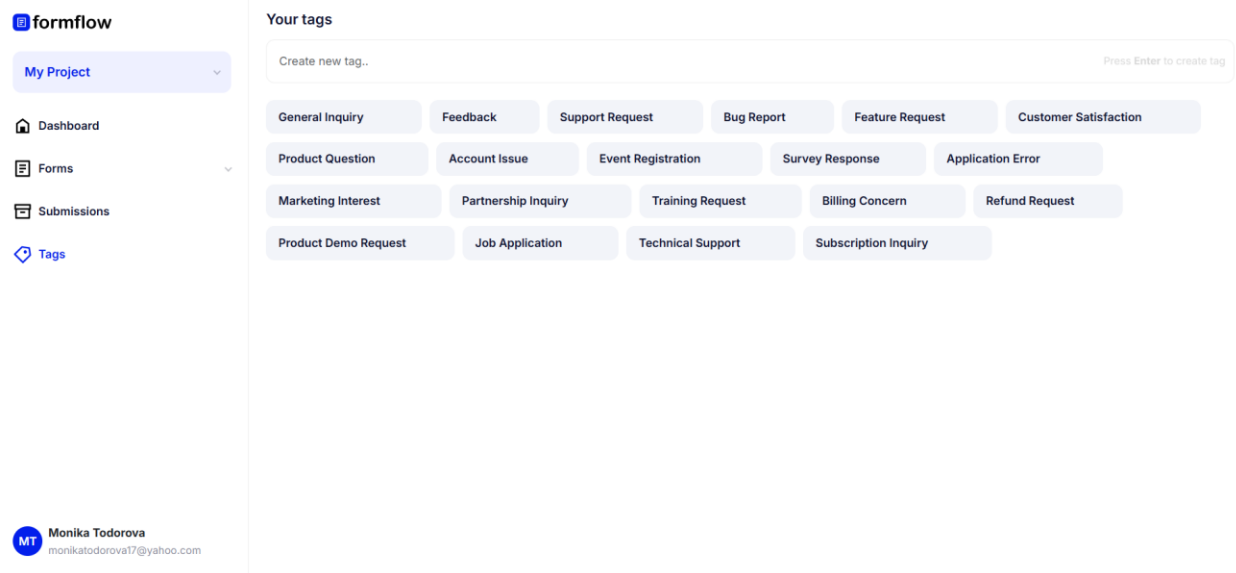
The screenshot shows a Microsoft Excel spreadsheet with the following data:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	name	email	phone	city	country									
2	Miss Dayne	pfannerstil	-9128	Wildermar	Korea									
3	Mr. Webster	oankundin	-8377	East Cryste	Senegal									
4	Donnell Wi	asa.roob@	(312) 809-	New Esme	Israel									
5	Dr. Martine	hamill.tore	-9113	South Edw	Turks and Caicos Islands									
6	Freedra Jon	eshanahar	+1 (928) 55	Breitenber	Libyan Arab Jamahiriya									
7	Delbert He	sprohaska	224-203-2	Port Jaysor	Aruba									
8	Alexanne N	grant.pour	+1 (541) 75	West Emm	San Marino									
9	Savanna Si	pherman@	442.790.8	North Dani	Philippines									
10	Joel Bayer	peichman	469.564.6	Alisonfort	Panama									
11	Else Vande	king75@vc	-6190	Port Patier	Taiwan									
12														
13														

Слика 60. Приказ на преземаниот документ

5.7 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 61. Екран за додавање и менаџирање на тагови

6. Заклучок

Оваа апликација претставува напредно решение кое на корисниците им овозможува едноставен и интуитивен начин за креирање, управување и користење на онлајн форми. Таа е дизајнирана да го олесни процесот на собирање на податоци и иницирање на комуникација преку дигитални канали, нудејќи практични алатки кои значително го подобруваат корисничкото искуство. Преку имплементација на функционалност за креирање на форми со форм билдер, корисниците можат без напор да ги прилагодат формите на своите специфични потреби. Дополнително, воведувањето на вештачка интелигенција како дел од системот го автоматизира процесот на генерирање полиња за формите, што ја зголемува ефикасноста и ја намалува потребата за техничко познавање.

Со користење на форм билдер, корисниците можат интуитивно да поставуваат полиња, дефинираат параметри и создаваат сложени форми без потреба од пишување на код. Од друга страна, вештачката интелигенција ги користи описите дадени од корисниците за автоматско генерирање на структурата на формите, овозможувајќи побрзо и поефикасно поставување на формите во системот. Оваа комбинација на флексибилност и автоматизација овозможува максимална продуктивност за корисниците.

Со оваа апликација, значително се поедноставува процесот на прибирање информации, а корисниците добиваат можност да се фокусираат на анализа на податоците наместо на техничките аспекти на формите. На овој начин, апликацијата не само што обезбедува практични алатки, туку и поставува стандарди за тоа како современите технологии, како што се вештачката интелигенција и модуларниот пристап, можат да се користат за решавање на секојдневни проблеми.

7. Референци

[1] – *Laravel documentation* <https://laravel.com/docs/10.x/releases>

[2] – *Vue JS*, <https://vuejs.org/>

[3] – *MySQL* <https://www.mysql.com>

[4] – *Eloquent*: <https://laravel.com/docs/10.x/eloquent>

[5] – *Vue JS Component*, <https://vuejs.org/guide/quick-start.html>