**ПРИДНІПРОВСЬКА ДЕРЖАВНА АКАДЕМІЯ БУДІВНИЦТВА ТА АРХІТЕКТУРИ**

Факультет інформаційних технологій та механічної інженерії

(повне найменування інституту, фікультету)

Кафедра Комп’ютерних наук, інформаційних технологій та прикладної математики

(повна назва кафедри)

**КВАЛІФІКАЦІЙНА РОБОТА**

на тему Розробка корпоративного веб-додатку для фіксації та оповіщення робітниками надзвичайних подій

Виконав: здобувач вищої освіти,   
Бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ступінь вищої освіти)

спеціальності

122 Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_

(шифр і назва спеціальності)

освітньої програми

Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_\_

(вид та назва ОП)

групи КН-17

Данило Орловський\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

Керівник Ілля Ільєв\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

Рецензент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

Оцінка захисту кваліфікаційної роботи (проекту)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(сума балів, оцінка ЄТКС, оцінка за національною шкалою)

Секретар ЕК \_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ім’я та прізвище)

Дніпро – 2021

**ПРИДНІПРОВСЬКА ДЕРЖАВНА АКАДЕМІЯ БУДІВНИЦТВА ТА АРХІТЕКТУРИ**

Інститут, факультет Інформаційних технологій та механічної інженерії\_\_\_\_

Кафедра Комп’ютерних наук, інформаційних технологій та прикладної математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рівень вищої освіти Бакалавр\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

Спеціальність Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

Освітня програма Комп’ютерні науки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва)

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри \_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

“\_\_\_\_\_” \_\_\_\_\_\_\_\_\_\_ 20\_\_\_ року

**З А В Д А Н Н Я**

**ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ**

**ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Орловського Данила\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я та прізвище)

1. Тема роботи Розробка корпоративного веб-додатку для фіксації та оповіщення робітниками надзвичайних подій\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи Ілля Ільєв, к.т.н., доцент кафедри комп’ютерних наук, інформаційних технологій та прикладної математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ім’я, прізвище, ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “08” лютого 2021 року №48-КС  
2. Строк подання роботи 14 червня 2021 року\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Ім’я, прізвище та посада  консультанта | Підпис, дата | |
| завдання видав | завдання  прийняв |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

7. Дата видачі завдання 01 лютого 2021 року \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів кваліфікаційної  роботи | Строк виконання етапів роботи | Примітка |
| 1 | Огляд літератури | 05.02.2021 – 22.03.2021 |  |
| 2 | Вивчення задач | 23.03.2021 – 16.04.2021 |  |
| 3 | Проектування парковки | 17.04.2021 – 04.05.2021 |  |
| 4 | Створення анімації | 05.05.2021 – 10.05.2021 |  |
| 5 | Оформлення пояснювальної записки | 11.05.2021 – 30.05.2021 |  |

**Здобувач вищої освіти \_\_\_\_\_\_\_** Данило Орловський

(підпис) (ім’я тпрізвище)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Ілля Ільєв\_\_\_\_\_\_

(підпис) (ім’я та прізвище)

**РЕФЕРАТ**

Пояснювальна записка складається з 97 сторінок та містить 39 рисунків. Перелік джерел посилання включає 8 найменувань.

Об’єкт дослідження – застосування фреймворків для розробки Веб-додатоку для фіксації та оповіщення робітників о надзвичайних подіях.

Мета роботи – розробити корпоративний Веб-додаток для фіксації та оповіщення робітників о надзвичайних подіях.

Методи дослідження – огляд існуючих рішень, для фіксування надзвичайних подій, розробка концептів форм для заповнення данних о подіях, розробка алгоритму фіксації та сповіщення.

У РОЗДІЛІ 1 знаходяться визначення та поняття стосовно веб-технологій та веб-программування.

У РОЗДІЛІ 2 описані популярні фрейморки для Web-розробки, визначені переваги кожного з фреймворків, та обгрунтовуеться вибір Фреймворку для робробки Веб-застосунка.

У РОЗДІЛІ 3 виконана розробка Веб-застосунку, наведена характеристика використаних інструментів для створення Веб-застосунку.

Ключові слова: Web-технології, програмування, Веб-додаток Front-end, Back-end, JavaScript, PHP, фреймворк.

**ЗМІСТ**

[ВСТУП 5](#_Toc74860363)

[Розділ 1. Web-технології 6](#_Toc74860365)

[1.1 Введення в Web-технології 6](#_Toc74860366)

[1.2 HTML 6](#_Toc74860367)

[1.3 CSS (SCSS) 7](#_Toc74860368)

[1.4 JavaScript 9](#_Toc74860369)

[1.5 HTTP 10](#_Toc74860370)

[1.6 PWA 12](#_Toc74860371)

[Розділ 2. Фреймворки 14](#_Toc74860372)

[2.1.1 JS-Фреймворки 14](#_Toc74860373)

[2.1.2 React 14](#_Toc74860374)

[2.1.3 Angular.js 16](#_Toc74860375)

[2.1.4 Vue.js 19](#_Toc74860376)

[2.2 PHP фрейморки 21](#_Toc74860377)

[2.2.1 Laravel 22](#_Toc74860378)

[2.2.2 Yii2 Framework 23](#_Toc74860379)

[2.2.3 Symphony Framework 25](#_Toc74860380)

[Розділ 3. Розробка Веб-застосунку 26](#_Toc74860381)

[3.1.1 Установка Laravel за допомогою функції create-project 26](#_Toc74860382)

[менеджера пакетів Composer. 26](#_Toc74860383)

[3.1.2 Міграції БД. 33](#_Toc74860384)

[3.1.3 Моделі Eloquent 34](#_Toc74860385)

[3.1.4 Маршрутизація 35](#_Toc74860386)

[3.2.1 Установка Vue.js 38](#_Toc74860387)

[3.2.2 Написання Vue додатку 39](#_Toc74860388)

[3.2.3 Html Drag 'n' Drop 40](#_Toc74860389)

[3.2.4 Сторінка Конструктора Форм 41](#_Toc74860390)

[4. Список літератури, що використовувалась: 59](#_Toc74860391)

[Додаток 1 59](#_Toc74860392)

[Додаток 2 63](#_Toc74860393)

[Додаток 3 68](#_Toc74860394)

[Додаток 4 75](#_Toc74860395)

[Додаток 5 99](#_Toc74860396)

# 

# ВСТУП

Цифрові технології охоплюють все більше різних сфер бізнесу. Для того щоб бізнесу залишатися конкурентоспроможним, звичайних засобів для оповіщення та опубліковування інформації вже недостатньо, необхідні мобільні web-застосунки, які надають користовучам не тільки інформуцію, а ще й дозволяють виконувати якісь функції.

Так, наприклад, сучасним бізнесам у сфері виробництва або у сфері надання послуг, для найбільш зручного контролю виконання якихось робіт або для більш швиткого реагування на надзвичані події необхідні мобільні Веб-застосунки у яких персонал сможе заповнивши швидку форму, оповістити о якійсь надзвичаній події відповідальних або оповістити керівництво о виконаній роботі.

Самі власники бізнесу, або відповідальні люди зможуть своєчасно отримати інформацію, переглянути статистику або обьем виконаних робіт за певний період часу.

Актуальність теми зумовлює те, що технології дуже стрімко розвиваються, паперові документи з кожним роком все більше і більше зникають з нашого життя. Та цей Веб-додаток дає можливість маючи тільки Смартфон заповнювати всі необхідні данні та зберігати іх на серверах без безкінечних паперців, будь-які данні неможливо будет десь загубити, зціми данними будет зручно іх опрацьовувати в подальщій роботі.

Мета - розробити корпоративний Веб-додаток для фіксації та оповіщення робітників о надзвичайних подіях.

# Розділ 1. Web-технології

# 1.1 Введення в Web-технології

Під «веб-технологіями» зазвичай маються на увазі мови розмітки, мови програмування, бази даних, системи і платформи CMS і інші технології, які дозволяють створювати веб-сайти, програми та магазини.

Є два типи веб-технологій. Перший, так званий **Front-end** - це той, який відповідає за зовнішній вигляд сайту, і видно в браузері користувача. Такі технології включають мову розмітки HTML, формальну мову опису зовнішнього вигляду документа CSS та мову программування JavaScript. Другий тип веб-технологій - **Back-end**, який працює на сервері і використовується для обробки даних. Їх дія зазвичай невидима для користувача, видно тільки введені дані або дії, виконані на веб-сайті, і результат цих даних або дій. Таким чином, весь процес виконання виходить за рамки браузера.

# 1.2 HTML

**HTML** (HyperText Markup Language - «мова гіпертекстової розмітки») - самий базовий будівельний блок Веб-сторінки. Він визначає зміст і структуру веб-контенту. Інші технології, крім HTML, зазвичай використовуються для опису зовнішнього вигляду / уявлення (CSS) або функціональності / поведінки (JavaScript) веб-сторінки. Під гіпертекстом розуміються посилання, які з'єднують веб-сторінки один з одним або в межах одного веб-сайту, або між веб-сайтами. Посилання є фундаментальним аспектом Інтернету. Завантажуючи контент в Інтернет і пов'язуючи його зі сторінками, створеними іншими людьми, ви стаєте активним учасником Всесвітньої павутини. HTML використовує розмітку "markup" для відображення тексту, зображень та іншої інформації в веб-браузері.

**HTML-розмітка** включає в себе спеціальні "елементи", такі як <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, < span>, <img>, <aside>, <audio>, <canvas>, <datalist »,« details »,« embed>, <nav>, <output>, <progress »,« video> і багато інших.

**HTML-елемент** виділяється з іншого тексту в документі за допомогою "тегів", які складаються з імені елемента оточеного "<" і ">". Ім'я елемента всередині тегу не чутливе до регістру. Тобто, воно може бути написано в верхньому або нижньому регістрі, або змішано. Наприклад, тег <title> може бути записаний як <Title>, <TITLE>, або будь-яким іншим способом.

# 1.3 CSS (SCSS)

**CSS** - це формальна мова, службовець для опису оформлення зовнішнього вигляду документа, створеного з використанням мови розмітки (HTML, XHTML, XML). Назва походить від англійського Cascading Style Sheets, що означає «каскадні таблиці стилів».

**Навіщо використовується CSS**

**Призначення CSS** - відокремлювати те, що задає зовнішній вигляд сторінки, від її змісту. Якщо документ створено тільки з використанням HTML, то в ньому визначається не тільки кожен елемент, але і спосіб його відображення (колір, шрифт, положення блоку і т.д.). Якщо ж підключені каскадні таблиці стилів, то HTML описує тільки черговість об'єктів. А за все їх властивості відповідає CSS. В HTML досить прописувати клас, не перераховуючи всі стилі кожен раз.

Така технологія:

* забезпечує відносно просту і швидку розробку, тому що одного разу створене оформлення можна застосовувати до багатьох сторінок;
* підвищує гнучкість і зручність редагування - досить внести правку в CSS, щоб оформлення змінилося всюди;
* робить код більш простим, знижуючи повторюваність елементів. Його простіше читати програмістам і пошуковим роботам;
* прискорює час завантаження, тому що CSS може кешуватися при першому відкритті, а в наступних зчитуються тільки структура і дані;
* збільшує кількість візуальних рішень для подання вмісту;
* забезпечує можливість легко застосовувати до одного документу різні стилі (наприклад, створювати адаптовану версію для мобільних пристроїв або спеціальні стилі для людей з вадами зору).

Тобто каскадні таблиці служать не тільки для втілення дизайну, а й кардинально змінюють підхід до сайтобудування, спрощуючи працю розробників і забезпечуючи гнучкість реалізації. Ось для чого потрібен CSS.

**SCSS** – “діалект” мови Sass, а **Sass** - це препроцесорнa мова, що базується на скриптах мови CSS. Використовується дана мова для того, щоб взаємодіяти зі стилями. Мета застосування: спростити і структурувати код, а також зробити програмування більш зручним.

Перелік головних переваг:

* можливість працювати з будь-якими параметрами;
* побудова таблиць і мереж;
* створення і взаємодія з вкладеними селекторами;
* верстка сайтів здійснюється із застосуванням додаткових констант і домішок СSS;
* можливість використання різних заготовок для спрощення коду;
* відмінне поєднання з різними системами автоматизації;
* велика кількість додаткових бібліотек;
* комфортну взаємодію з вкладеними спектрами.

Незначні недоліки:

* збільшення обсягу коду;
* низький спектр завдань для прискорення;
* ймовірні проблеми з циклічністю.

# 1.4 JavaScript

JavaScript (частіше просто JS) - це легкий, що інтерпретується або JIT-компільований, об'єктно-орієнтована мова з функціями першого класу. Найбільш широке застосування знаходить як мова сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад, node.js або Apache CouchDB. JavaScript - це прототипно-орієнтована мова з динамічною типізацією, яка підтримує об'єктно-орієнтований стиль програмування.

**Прототипно-орієнтований стиль** об'єктно-орієнтованого програмування той, в якому класи визначені неявно, а похідні класи виходять додаванням властивостей і методів в екземпляр іншого класу або, значно рідше, до порожнього об'єкту.

Якщо мова програмування має **функції першого класу**, то значить вони трактуються як об'єкти, тобто можуть бути передані іншим функціям і їх можна повернути з функцій. Так само їх можна привласнювати змінним.

Стандартом мови JavaScript є **ECMAScript**. Всі сучасні браузери повністю підтримують ECMAScript 5.1. Більш ранні версії браузерів підтримують принаймні - ECMAScript 3. 17 червня 2015 року відбувся випуск шостої версії ECMAScript. Ця версія офіційно називається ECMAScript 2015 року, яку найчастіше називають ECMAScript 2015 або просто ES2015. З недавнього часу стандарти ECMAScript випускаються щорічно.

# 1.5 HTTP

**HTTP** - широко поширений протокол передачі даних, спочатку призначений для передачі гіпертекстових документів (тобто документів, які можуть містити посилання, що дозволяють організувати перехід до інших документів).

Абревіатура HTTP розшифровується як **HyperText Transfer Protocol**, «протокол передачі гіпертексту». Відповідно до специфікації OSI, HTTP є протоколом прикладного (верхнього, 7-го) рівня. Актуальна на даний момент версія протоколу, HTTP 1.1, описана в специфікації RFC 2616.

Протокол HTTP припускає використання клієнт-серверної структури передачі даних. Клієнтську програму формує запит і відправляє його на сервер, після чого серверне програмне забезпечення обробляє цей запит, формує відповідь і передає його назад клієнтові. Після цього клієнтську програму може продовжити відправляти інші запити, які будуть оброблені аналогічним чином.

Завдання, яке традиційно вирішується за допомогою протоколу HTTP - обмін даними між призначеним для користувача додатком, що здійснює доступ до веб-ресурсів (зазвичай це веб-браузер) і веб-сервером. На даний момент саме завдяки протоколу HTTP забезпечується робота Всесвітньої павутини.

Також HTTP часто використовується як протокол передачі інформації для інших протоколів прикладного рівня, таких як SOAP, XML-RPC і WebDAV. У такому випадку говорять, що протокол HTTP використовується як «транспорт».

API багатьох програмних продуктів також має на увазі використання HTTP для передачі даних - самі дані при цьому можуть мати будь-який формат, наприклад, XML або JSON.

Як правило, передача даних по протоколу HTTP здійснюється через TCP / IP-з'єднання. Серверне програмне забезпечення при цьому зазвичай використовує **TCP-порт** 80 (і, якщо порт не вказано явно, то зазвичай клієнтське програмне забезпечення за замовчуванням використовує саме 80-й порт для відкритих HTTP-з'єднань), хоча може використовувати і будь-який інший.

**HTTPS** (від англ. HyperText Transfer Protocol Secure - безпечний протокол передачі гіпертексту) - це розширення протоколу HTTP, що підтримує шифрування за допомогою криптографічних протоколів SSL і TLS.

Чим відрізняються HTTP від HTTPS

* HTTPS не є окремим протоколом передачі даних, а являє собою розширення протоколу HTTP з надбудовою шифрування;
* передаються по протоколу HTTP дані не захищені, HTTPS забезпечує конфіденційність інформації шляхом її шифрування;
* HTTP використовує порт 80, HTTPS - порт 443.

# 1.6 PWA

**PWA** (Progressive Web App) - це технологія в веб-розробці, з її допомогою можна побудувати сайт візуально і функціонально нагадує мобільний додаток, тільки лише він буде відображатися в браузері.

PWA сайти будуються за допомогою HTML, JavaScript, CSS, і в браузері виглядають як звичайний сайт, але взаємодіють з відвідувачем як мобільний додаток. Розширений функціонал дозволяє додавати сайт або окрему його сторінку на екран будь-яких пристроїв, а також відправляти push-повідомлення.

PWA дає можливість сайту-додатком працювати в автономному режимі.

**Переваги та недоліки PWA**

PWA має такі **переваги**:

* PWA можна завантажити на будь-який мобільний пристрій (телефон, планшет), десктоп, ноутбук. Також технологія підтримується практично всіма популярними операційними системами.
* Користувачам не треба завантажувати та встановлювати оновлення самостійно, новий функціонал і будь-які зміни додаються розробниками віддалено.
* Service Worker і стратегія кешування дає офлайн доступ до ресурсу.
* PWA легше і дешевше розробити, ніж створити нативное додаток під кожну операційну систему.
* Progressive Web Apps набагато менше за розміром, ніж нативні додатки, тому що прогресивні додатки ефективно використовують можливості браузера. При цьому установка відбувається миттєво, так як всі необхідні компоненти вже були встановлені в кеш при першому торканні користувача з сайтом.
* З 2019 року PWA можна встановити як минаючи магазини додатків, так і через Google Play, App Store, і т.д.
* Можливість переходу на PWA з посилань в соцмережах, з видачі, перебуваючи на сайті.
* Істотна економія і свобода від правил магазинних додатків, до того ж вам не треба платити до 30% від обсягу продажів.
* Як і нативное додаток, PWA має свою власну оболонку, тобто запускається окремо від браузера після установки.
* Pop-up повідомлення, робота в автономному режимі, а також весь інший функціонал PWA працюватимуть і без установки програми.

На даний момент PWA - це досить нова й поки ще не універсальна технологія, яка має такі **недоліки**:

* Деякі пристрої, браузери і ОС підтримують частковий функціонал PWA.
* Немає можливості налагодити повне і активне використання iOS, при цьому версія системи повинна бути 11.3+. Також є обмеження до доступу деяких вбудованих функцій як Siri, In-App Payments, і інше.
* Робота та функціональність сайту в офлайн режимі обмежена.
* Варто враховувати, що використання PWA значно збільшує витрату заряду батареї.

# Розділ 2. Фреймворки

# 2.1.1 JS-Фреймворки

**Фреймворки JS** - це бібліотеки програмування JavaScript, в яких є попередньо написаний код для використання в стандартних функціях і завданнях програмування. Це основа для створення веб-сайтів або веб-додатків навколо.

Використання фреймворків в першу чергу підвищить вашу продуктивність. Тому, що доведеться писати менше коду вручну, так як, вже є заздалегідь написані і готові до використання функції і шаблони. Деякі компоненти веб-сайту не повинні бути виготовлені за індивідуальним замовленням, тому ви можете створювати і розширювати попередньо створені компоненти. Фреймворки більш адаптованості для дизайну веб-сайтів, і більшість розробників сайтів вважають за краще їх.

# 2.1.2 React

React - JavaScript-бібліотека для роботи з користувача інтерфейсамі (UI), якові создали розробник Facebook. Бібліотеку начали використовуват на сайті цієї соціальної мережі в 2011 году. А в 2013 году Facebook відкрів вихідний код React.

С помощью React розробник створюють веб-Додатки, Які змінюють відображення без перезавантаження Сторінки. Завдяк цьом додатка Швидко реагують на Дії користувача, например, Заповнення форм,! Застосування фільтрів, Додавання товарів в кошик и так далі.

React застосовують для відтворення компонентів для користувача інтерфейсу. Такоже бібліотека может Повністю управляти фронтенда. У цьому випадку React Використовують з бібліотекамі для управління станом и роутінга, например, Redux и React Router.

**Особливості React**

1. **Односпрямована передача даних**

Властивості передаються від батьківських компонентів до дочірнім. Компоненти отримують властивості як безліч незмінних (англ. Immutable) значень, тому компонент не може безпосередньо змінювати властивості, але може викликати зміни через callback-функції. Такий механізм називають «властивості вниз, події наверх».

1. **Віртуальний DOM**

React використовує віртуальний DOM (англ. Virtual DOM). React створює кеш-структуру в пам'яті, що дозволяє обчислювати різницю між попереднім і поточним станами інтерфейсу для оптимального поновлення DOM браузера. Таким чином програміст може працювати зі сторінкою, вважаючи, що вона оновлюється вся, але бібліотека самостійно вирішує, які компоненти сторінки необхідно оновити.

1. **JSX**

JavaScript XML (JSX) - це розширення синтаксису JavaScript, яке дозволяє використовувати HTML-подібний синтаксис для опису структури інтерфейсу. Як правило, компоненти написані з використанням JSX, але також є можливість використання звичайного JavaScript. JSX нагадує іншу мову, створений в компанії Фейсбук для розширення PHP, XHP.

1. **Методи життєвого циклу**

Методи життєвого циклу дозволяють розробнику запускати код на різних стадіях життєвого циклу компонента. наприклад:

***shouldComponentUpdate*** - дозволяє запобігти перерисовку компонента за допомогою повернення false, якщо перерисовка не потрібна.

***componentDidMount*** - викликається після першої відтворення компонента. Часто використовується для запуску отримання даних з віддаленого джерела через API.

***render*** - найважливіший метод життєвого циклу. Кожен компонент повинен мати цей метод. Зазвичай викликається при зміні даних компонента для перемальовування даних в інтерфейсі.

1. **Не тільки отрисовка HTML в браузері**

React використовується не тільки для відтворення HTML в браузері. Наприклад, Facebook має динамічні графіки, які отрісовиваємих в теги <canvas>. Netflix і PayPal використовують ізоморфні завантаження для відтворення ідентичного HTML на сервері і клієнті.

1. **React Hooks**

Хукі дозволяють використовувати стан і інші можливості React без написання класів. Побудова призначених для користувача хуков дозволяє поміщати логіку компонента в повторно використовуваних функцій.

# 2.1.3 Angular.js

**AngularJS** - структурований фреймворк для динамічних web-додатків. Дозволяє використовувати HTML в якості мови шаблонів, а так же розширювати HTML-синтаксис, щоб код вашої програми виглядав коротко і лаконічно. Використовуючи Data-binding і Dependency injection (що йдуть з коробки), ви також зменшуєте кількість свого коду. І все це на браузерному JavaScript і працює з будь-серверної технологією.

Angular - це те, чим був би HTML, будь він призначений для додатків. HTML - чудовий декларативний мову для статичних документів. Він не містить багато чого для створення додатків, і, як результат, побудова в ньому веб-додатків зводиться до того щоб обдурити браузер, змусивши його робити те, що я хочу.

Невідповідності між динамічними додатками і статичними документами часто вирішуються наступними способами:

* бібліотеки - колекції функцій, які часто використовуються при створенні веб-додатків. Ви розширюєте свій код і викликаєте функції з бібліотеки, коли необхідно. Наприклад, jQuery;
* фреймворки - певні реалізації веб-додатків, де ваш код деталізує логіку. Фреймворк бере на себе управління і викликає ваш код в тих випадках, коли йому необхідно зробити щось специфічне для додатка. Приклади: knockout, sproutcore і т. Д.

Angular пропонує інший підхід. Він намагається звести до мінімуму невідповідність між документо-центричним HTML і потребами додатки шляхом створення нових HTML-конструкцій. Angular навчає браузер новому синтаксису за допомогою конструкцій, які називаються директивами. Можна навести такі приклади:

* Біндінг даних у вигляді {{}}.
* Управління структурами DOM для повторюваних / приховуваних фрагментів DOM.
* Підтримка форм і перевірок форм.
* Зв'язування коду з DOM-елементами.
* Угруповання HTML в повторно використовувані компонент

Angular позиціонується як закінчене рішення в розробці веб-додатків. Це означає, що він є не просто частина загальної головоломки побудови веб-додатки, а є цілим комплексом. Це дає право Angular вказувати на те, як повинні будуватися CRUD-додатки. Але в той же час він намагається переконатися, що його думка з цього приводу - лише відправна точка, яку Ви можете легко змінити. З коробки Angular включає:

* Все, що потрібно при побудові CRUD-додатки - це зв'язка з прив'язки даних, основних директив шаблонів, валідації форм, маршрутизації, deep-linking, повторно застосовуваних компонентів і впровадження залежностей.
* Тестування: модульні (unit) тести, end-to-end тести, Моки, test harnesses.
* Початкове додаток до структури директорій і скриптом тестування в якості відправної точки.

**Область застосування Angular**

Angular спрощує розробку додатків надаючи розробнику більш високий рівень абстракції. Як і за будь-яку іншу абстракцію, за цю доводиться платити гнучкістю. Іншими словами, Angular підійде не для кожної програми. Angular призначений для CRUD-додатків. На щастя, CRUD-додатки складають як мінімум 90% всіх веб-додатків. Але важливо розуміти, коли Angular буде хорошим вибором, а коли не дуже.

Ігри та редактори інтерфейсу - яскраві приклади дуже інтенсивних і складних маніпуляцій з DOM. Ці види додатків відрізняються від CRUD-додатків, і, як результат, не підходять для Angular. В цьому випадку щось більш близьке до заліза, як, наприклад, jQuery, може бути набагато кращим варіантом.

# 2.1.4 Vue.js

Vue (вимовляється / vjuː /, приблизно як view) - це прогресивний фреймворк для створення користувацьких інтерфейсів. На відміну від фреймворків-монолітів, Vue створений придатним для поступового впровадження. Його ядро в першу чергу вирішує завдання рівня уявлення (view), що спрощує інтеграцію з іншими бібліотеками та існуючими проектами. З іншого боку, Vue повністю підходить і для створення складних односторінкових додатків (SPA, Single-Page Applications), якщо використовувати його спільно з сучасними інструментами та додатковими бібліотеками.

**Концепція**

Vue.js називають прогресивним і поступово адаптованим в порівнянні з іншими веб-фреймворками.

Це дозволяє розробнику налаштувати структуру програми відповідно до власних вимог. Vue.js вважають простішим в освоєнні, ніж AngularJS, оскільки API побудований набагато простіше в освоєнні. У Vue.js можна використовувати тільки знання JavaScript і HTML. Можливе застосування Typescript. У Vue.js є власна офіційна досить багата документація на багатьох мовах, викладена на vuejs.org, яка може послужити прикладом в поясненні проектування і розробки в браузері. У Vue.js реалізується шаблон MVVM, Vue.js пропонує можливість прив'язки даних на Javascript, так що висновок і введення даних сполучаються безпосередньо з джерелом даних. Таким чином, режим ручного визначення даних (наприклад, через jQuery) з HTML-DOM не потрібен. При цьому немає необхідності в жодних додаткових анотаціях, як в Knockout.js, оголошені в Vue-Element звичайні змінні JavaScript включаються в якості реактивних елементів.

**Патерн MVVM** (Model-View-ViewModel) дозволяє відокремити логіку додатки від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.

Даний патерн був представлений Джоном Госсманом (John Gossman) в 2005 році як модифікація шаблону Presentation Model і був спочатку націлений на розробку додатків в WPF. І хоча зараз цей патерн вийшов за межі WPF і застосовується в самих різних технологіях, в тому числі при розробці під Android, iOS, проте WPF є досить показовою технологією, яка розкриває можливості даного патерну.

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View).

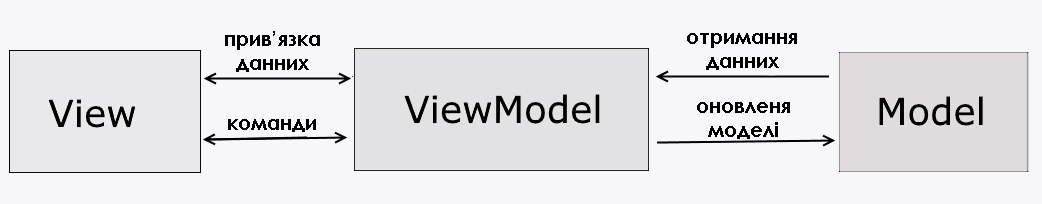


Рис. 2.1. Схема патерну MVVM

**Адаптивний дизайн**

Чуйний означає, що подання до моделі MVC змінюється в міру зміни моделі. В Vue розробники просто прив'язують уявлення до відповідної моделі, і Vue автоматично спостерігає за змінами в моделі і перемальовує уявлення. Ця функція робить управління станом Vue досить простим і інтуїтивно зрозумілим.

**Ефекти переходу**

Vue надає різні способи застосування ефектів переходу при вставці, оновленні або видаленні DOM.  
Включает наступні інструменти:

* Автоматично застосовувати CSS класи при переходах і анімації
* Ви можете працювати зі сторонніми бібліотеками анімації CSS
* Використовуйте JavaScript функції для перехоплення переходу, щоб безпосередньо керувати DOM
* Може використовуватися в поєднанні зі сторонніми бібліотеками анімації JavaScript

# 2.2 PHP фрейморки

**PHP** - один з найпопулярніших і затребуваних мов програмування. Його активно використовують великі проекти, наприклад, Facebook і Wikipedia. На PHP написані популярні системи управління контентом (CMS), в тому числі WordPress. На цьому движку працює близько третини всіх сайтів в інтернеті і близько 60% сайтів на CMS.

PHP розвивається, а версія 7 зробила цю мову стабільним. В такому випадку, навіщо потрібні фреймворки і чому їх часто використовують замість нативного мови? Ось кілька причин:

* Фреймворки прискорюють розробку. Наприклад, PHP-фреймворк позбавляє вас від необхідності писати запити до баз даних. У фреймворк реалізовані базові функції CRUD, які необхідні для роботи з базами даних.
* Написані на фреймворками додатки легко масштабуються.
* Підтримувати легше проект на фреймворку, ніж на нативном PHP. Код фреймворків лаконічний, тому з ним просто працювати.
* В PHP-фреймворк використовується модель MVC. Вона спрощує розробку.
* Додатки на фреймворками краще захищені, ніж додатки на чистому PHP.
* У фреймворк реалізований принцип DRY. Це дозволяє розробникам писати менше коду.

Нативний PHP дозволяє робити додатки. Але перерахованих вище переваг досить, щоб звернути увагу на фреймворки.

# 2.2.1 Laravel

**Laravel** - це фреймворк для web-додатків з виразним і елегантним синтаксисом. Він дозволить спростити вирішення основних наболілих завдань, таких як аутентифікація, маршрутизація, сесії і кешування. Laravel - це спроба об'єднати все найкраще, що є в інших PHP фреймворк. Основні переваги Laravel:

* Laravel підтримує функціональне, інтеграційне і юніт-тестування
* При розробці корпоративного програмного забезпечення дуже важливо мати можливість виконувати тести. Від цього залежить якість продукту. При роботі з Laravel можна обійтися без сторонніх інструментів, так як фреймворк нативної підтримує тестування компонентів.
* Створені на Laravel додатки легко масштабуються
* Коли необхідно масштабувати додаток на Laravel, ви просто берете і масштабіруете його. У фреймворку є нативні інструменти, наприклад, Laravel Cache для кешування, Laravel Horizon для організації черг та інші служби, які можуть використовувати можливості хмарних сервісів. Проект на Laravel можна інтегрувати з GitHub, CircleCI, Docker, Amazon Web Services і так далі.
* У Laravel розвинена екосистема інструментів, які можна використовувати для роботи з проектами ентерпрайз-класу
* Laravel - проект з відкритим вихідним кодом
* Laravel - на 100% опенсорсний проект. Це відкриває великі можливості для кастомізації, модифікації і розширення, і тут Laravel перевершує багато популярні фреймворки.
* Laravel слід кращим практикам розробки і забезпечує великий вибір шаблонів проектування

Незважаючи на свою молодість (перший реліз вийшов в 2011 році), це вже зовсім зрілий продукт, і, згідно з опитуванням, проведеним порталом SitePoint, він посідає перше місце за популярністю серед розробників на PHP.

# 2.2.2 Yii2 Framework

**Фреймворк Yii** - це сучасний PHP фреймворк для розробки веб-додатків будь-якого рівня складності. Yii, як і будь-який сучасний фреймворк, реалізує патерн MVC, а це значить, що логіка буде відділена від відображення, що зробить код чистішим і логічним.

Як і інші фреймворки, Yii пропонує нам величезну кількість додаткових можливостей, серед яких: безпека з коробки (валідація і фільтрація даних, захист від SQL-ін'єкцій і XSS), зручна робота з БД, Yii debug і багато інших булочок, що прискорюють роботу по створенню сайту.

**Переваги Yii.**

**Переваги Yii - це висока швидкість роботи і хороша підтримка OОП**. Yii включає в себе велику кількість бібліотек. Завдяки їм можна легко створити веб-додаток, яке відповідатиме всім сучасним стандартам. Вбудовані методи дозволяють значно скорочувати кількість коду.

Важливо, що **Yii не тільки поширюється безкоштовно, але і виходить під ліцензією BSD**. Тобто його можна використовувати в комерційних цілях, вбудовувати в пропрієтарні продукти.

**Yii має хорошу документацію**. Вона довгий час була англомовної, але силами російськомовних користувачів переклад все-таки побачив світ і тепер доступний на сайті проекту.

**Фреймворк Yii відомий своїм величезним співтовариством**, в тому числі російською мовою. Великі спільноти розробників - це можливість оперативно отримати допомогу і обговорити важливі теми. Більш того - на форумі можна поспілкуватися і з самим творцем YiiQiang Xue (писати йому варто англійською або китайською).

Проте Yii не можна назвати проектом однієї людини, так як зараз **фреймворк підтримує і розвиває велика команда**. Вона стежить за основними IT-тенденціями і впроваджує їх у проект.

**Yii має хороший генератор коду.** Він згенерує вихідний код, тобто структуру програми, в залежності від параметрів, які ви вкажете.

**Yii має гнучкий механізм кешування** і підтримує кілька сховищ для кешування змінних, ділянок коду і так далі.

Якщо ви не знайдете в Yii потрібні вам **класи**, то їх можна буде **підключити як розширення** з інших фреймворків.

**Yii - фреймворк має досить низький поріг входження**, однак, як і будь-який інший інструмент, він вимагає деякого часу для вивчення, щоб потім можна було використовувати його максимально ефективно і зручно.

**Філософія, якої дотримуються розробники фреймворка, - це простота і елегантність коду**. Yii добре розширюється, в ньому легко працювати з кодом: ви зможете замінити або відредагувати практично будь-яку частину основного коду, а також зможете поділитися (або скористатися) кодом, який надає співтовариство.

**Yii сприяє швидкому прототіпірованію проекту**. Це зручно на ранніх стадіях розробки, але не варто забувати, що головна мета прототипу - демонстрація працює моделі системи. Прототип - це не кінцевий продукт.

**Продуктивність** - це одна з основних цілей Yii. Тому за результатами тестів phpmark він має кращу продуктивність.

**Розробники Yii приділяють багато уваги безпеки фреймворка**, тому в Yii передбачені заходи проти SQL-ін'єкцій, міжсайтового скриптинга (XSS), фальсифікації файлів cookie та інших атак на веб-додатки.

# 2.2.3 Symphony Framework

**Symfony** - це набір багаторазово використовуваних компонентів PHP, що дозволяють розробнику створювати масштабовані високопродуктивні додатки. З **30 компонентами, з яких можна вибрати, розробник має повну свободу експериментувати і працювати** в середовищі RAD.

**API Symfony також забезпечують просту інтеграцію зі сторонніми додатками** і можуть використовуватися з популярними інтерфейсними фреймворками, такими як AngularJS.

Багато популярних проекти, включаючи Drupal і phpBB, також використовують фреймворк Symfony. Насправді, Laravel, найпопулярніший PHP-фреймворк, побудований на Symfony.

**Symfony працює на багаторазових компонентах** і забезпечує кращу модульність. Symfony також використовує модель і контролер для розробки веб-додатку, який може виглядати іржавим для багатьох нових розробників, але воно працює. Крім того, **Symfony є хорошим прикладом модульної структури**. Ви можете використовувати 30 компонентів забезпечених Symfony в вашому проекті в модульному способі.

**Symfony можна використовувати для швидкого розвитку і складних проектів.** Незважаючи на те, що існує дискусія про те, яка структура краще підходить для складних проектів , Symfony демонструє блискучу обробку складності в порівнянні з іншими структурами. Yii також використовує компоненти, але не так модульно, як Symfony. Laravel не забезпечує модульний підхід настільки ж гострий, як дві інші структури.

**Фреймворк symfony**: пропонує випуск LTS, поставляється з великою кількістю функцій, **в даний час є найбільш стабільною структурою**, компонент-заснована рамка і пропонує широку модульність, і має велике співтовариство з великою кількістю навчальних ресурсів.

# Розділ 3. Розробка Веб-додатку

# 3.1.1 Установка Laravel за допомогою функції create-project

# менеджера пакетів Composer.

Скористаемося функцією create-project менеджера пакетів Composer, яка дозволяє створювати проекти з певною структурою. Для цього виконаемо наступну команду:

composer create-project laravel / laravel projectName

У поточному каталозі також буде створений підкаталог з ім'ям {projectName} з попередніми каркасом додатка.

**Структура каталогів Laravel**

При відкритті каталогу з заготівлею додатки Laravel ви побачите наступніи файли і каталоги:

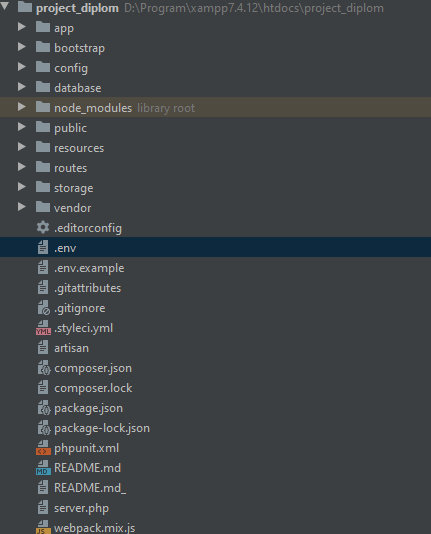


Рис. 3.1. Структура каталогів Laravel

**Каталоги.**

Кореневої каталог за замовчуванням містить наступні папки.

**app** - тут розміщується основна частина вашого застосування - моделі, контролери, команди і PHP-код домену.

**bootstrap** - містить файли, які Laravel використовує для завантаження при щоразу під час запуску.

**config** - тут знаходяться всі конфігураційні файли.

**database** - містить міграції баз даних, сідери і фабрики.

**node\_modules** – тут знаходяться інструменти менеджера пакетів NPM.

**public** - каталог, на який вказує сервер при обслуговуванні сайту. Містить файл index.php - фронтальний контролер, який запускає процес початкового завантаження і маршрутизує всі запити. Тут також розміщуються всі публічні файли: зображення, таблиці стилів, сценарії або компоненти для завантаження.

**resources** - тут знаходяться файли для інших сценаріїв: уявлення,

мовні файли, а також (опціонально) файли вихідного коду CSS / Sass і файли вихідного коду JavaScript.

**routes** - містить всі визначення маршрутів як для HTTP-маршрутів, так

і для «консольних маршрутів» або команд Artisan.

**storage** - тут знаходяться кеші, логи і скомпільовані системні файли.

**vendor** - сюди встановлюються залежно менеджера пакетів Composer.

Цей каталог ігнорується системою управління версіями Git (позначається як неконтрольований нею) в силу того, що дії Composer є складовою частиною процесу розгортання на будь-яких віддалених серверах.

**Окремі файли.**

Кореневої каталог також містить наступні файли.

**.editorconfig** - інструкції для вашої середовища розробки / текстового редактора щодо пропонуються фреймворком стандартів кодування (наприклад, про розмір відступів, кодуванні і про те, чи слід обрізати кінцеві пробіли).

**.env і .env.example** - задають змінні середовища (імовірно є різними в різних середовищах і тому не реєструються в системі управління

версіями).

**.env.example** - це шаблон, який дублюється кожної конкретної середовищем для створення власного файлу .env, ігнорованого системою управління версіями Git.

**.gitignore і .gitattributes** - конфігураційні файли системи управління версіями Git.

**artisan** - дозволяє запускати команди Artisan з командного строки.

**composer.json і composer.lock** - конфігураційні файли для Composer, при цьому файл composer.json може редагуватися користувачем, а файл composer.lock - немає. Містять деякі базові відомості про проект, а також визначають його PHP-залежності.

**package.json** - файл, аналогічний composer.json, але призначений для ресурсів клієнтської частини і залежностей системи збирання. містить вказівки для менеджера пакетів NPM щодо того, які залежності JavaScript слід довантажити.

**phpunit.xml** - конфігураційний файл для PHPUnit - інструменту, який Laravel використовує для тестування системи.

**readme.md** - файл Markdown, що містить базові відомості про фреймворку.

**server.php** - резервний сервер, що дозволяє виконувати попереднійперегляд програми Laravel навіть малопотужним серверів.

**webpack.mix.js** - конфігураційний (опціональний) файл для Mix. Містятить вказівки для системи збирання щодо способу компіляції та обробки ресурсів клієнтської частини.

**Конфігурація**

Основні налаштування додатка Laravel - налаштування підключення до бази даних, параметри обробки черг, електронної пошти і т. д. – містяться в файлах папки config. Кожен з цих файлів повертає масив мови PHP, доступ до елементів якого здійснюється по конфігураційному ключу, що складається з імені файлу і всіх ключів-нащадків, між якими ставлять крапку (.).

Так, створюю файл config / services.php, що містить код такого вигляду:

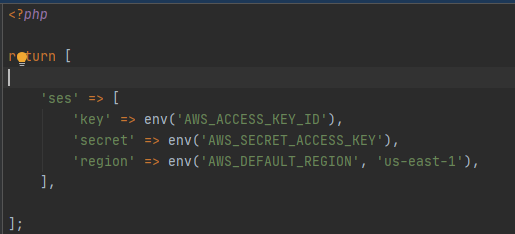


Рис. 3.2. Вміст файлу config/services.php

Будь-які змінні конфігурації, які повинні бути різними у різних середовищ (І ігноруватися системою управління версіями), потрібно перенести з цієї папки в файли .env.

Помічник env () витягує значення з файлу .env, що містить потрібний вам ключ. Відповідно, слід додати його в файл .env (який зберігає настройки даної середовища) і .env.example (є шаблоном для всіх середовищ):

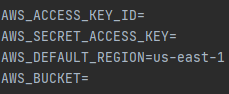


Рис. 3.3. Конфіграційний ключ

**Файл .env.**

Коротко розглянемо вміст файлу .env за замовчуванням.

APP\_NAME=Laravel  
APP\_ENV=local  
APP\_KEY=base64:iDilAD7RtXyJ+mePtlxU+j20RLJarcKvZZAKGkqDYxY=  
APP\_DEBUG=true  
APP\_URL=http://localhost  
  
LOG\_CHANNEL=stack  
  
DB\_CONNECTION=mysql  
DB\_HOST=127.0.0.1  
DB\_PORT=3306  
DB\_DATABASE=diplom\_db  
DB\_USERNAME=root  
DB\_PASSWORD=  
  
BROADCAST\_DRIVER=log  
CACHE\_DRIVER=file  
QUEUE\_CONNECTION=sync  
SESSION\_DRIVER=file  
SESSION\_LIFETIME=120  
  
REDIS\_HOST=127.0.0.1  
REDIS\_PASSWORD=null  
REDIS\_PORT=6379  
  
MAIL\_MAILER=smtp  
MAIL\_HOST=smtp.mailtrap.io  
MAIL\_PORT=2525  
MAIL\_USERNAME=null  
MAIL\_PASSWORD=null  
MAIL\_ENCRYPTION=null  
MAIL\_FROM\_ADDRESS=null  
MAIL\_FROM\_NAME="${APP\_NAME}"  
  
AWS\_ACCESS\_KEY\_ID=  
AWS\_SECRET\_ACCESS\_KEY=  
AWS\_DEFAULT\_REGION=us-east-1  
AWS\_BUCKET=  
  
PUSHER\_APP\_ID=  
PUSHER\_APP\_KEY=  
PUSHER\_APP\_SECRET=  
PUSHER\_APP\_CLUSTER=mt1  
  
MIX\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"  
MIX\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

Рис. 3.4. Вміст файлу .env

Варто звернути увагу на дві важливі змінні середовища:

**APP\_KEY** - випадково згенерувала рядок для шифрування даних. Якщо цей ключ буде порожнім, то можливо з’явиться помилка «Не вказаний ключ шифрування додатки ». Тоді треба запустити команду php artisan key: generate, і Laravel згенерує ключ для вас.

**APP\_DEBUG** - логічне значення, що визначає, чи повинні користувачі цього примірника вашого застосування бачити помилки налагодження, - добре підходить для локальних і проміжних середовищ і абсолютно не підходить для експлуатаційної.

Іншим не пов'язаним з аутентифікацією параметрам (BROADCAST\_DRIVER, QUEUE\_CONNECTION і т. Д.) присвоюються значення за замовчуванням, що забезпечують мінімально можливу залежність від зовнішніх сервісів.

Для більшості проектів після запуску програми потрібно змінити параметри конфігурації бази даних. Оскільки я використовую Laravel Valet, то привласнюю параметру DB\_DATABASE ім'я свого проекту, параметру DB\_USERNAME - значення root, а параметру DB\_PASSWORD - порожній рядок:

DB\_CONNECTION=mysql  
DB\_HOST=127.0.0.1  
DB\_PORT=3306  
DB\_DATABASE=diplom\_db  
DB\_USERNAME=root  
DB\_PASSWORD=

Рис. 3.4. Конфігурація бази даних

Потім я створюю базу даних з таким же, як у проекту, ім'ям в MySQL клієнті.

# 3.1.2 Міграції БД.

Створимо таблицю БД, яка буде містити всі форми сроблені користувачами. Для створення різних класів може бути використаний інтерфейс Artisan. Він позбавить нас від ручної генерації коду при створенні проектів Laravel. Тому давайте використаємо команду **make: migration** для створення міграції нової бази даних для нашої таблиці forms:

php artisan make: migration create\_form\_table --create=forms

Міграція буде поміщена в каталог database / migrations проекту. Команда *make: migration* додала автоінкремент ID і мітки часу до файлу міграції. Відредагуємо цей файл і додамо стовпці для id, id клієнта, данними форми та стопці для полей з мітками часу:

<?*php*use Illuminate\Database\Migrations\Migration;  
use Illuminate\Database\Schema\Blueprint;  
use Illuminate\Support\Facades\Schema;  
  
class CreateFormTable extends Migration  
{  
 /\*\*  
 \* Run the migrations.  
 \*  
 \* @return void  
 \*/  
 public function up()  
 {  
 Schema::create('forms', function (Blueprint $table) {  
 $table->id();  
 $table->integer('client\_id');  
 $table->json('data');  
 $table->timestamps();  
 });  
 }  
 /\*\*  
 \* Reverse the migrations.  
 \*  
 \* @return void  
 \*/  
 public function down()  
 {  
 Schema::dropIfExists('forms');  
 }  
}

Рис. 3.5. Вміст файлу міграції Laravel

**Щоб запустити нашу міграцію**, ми будемо використовувати команду Artisan Shmigrate

php artisan migrate

# 3.1.3 Моделі Eloquent

**Eloquent** - це стандартне ORM для Laravel (об'єктно-реляційне відображення). Eloquent робить безболісним отримання і зберігання даних в базі даних, використовуючи чітко визначені «моделі». Зазвичай, кожна Eloquent модель однозначно відповідає одній таблиці бази даних.

Визначимо **модель Form**, яка буде відповідати створеної таблиці forms. Ми знову можемо використовувати команду Artisan, щоб згенерувати цю модель. У цьому випадку ми будемо використовувати команду make: model:

php artisan make: model Form

Модель буде поміщена в каталог app. За замовчуванням клас моделі порожній. Нам не треба явно вказувати, який таблиці відповідає Eloquent модель, тому що мається на увазі, що ім'я таблиці - це ім'я моделі у множині (s на кінці). У цьому випадку модель Form, як передбачається, відповідає таблиці бази даних forms.

Код знаходиться у додатку 1

Таблиця forms буде виглядати так:

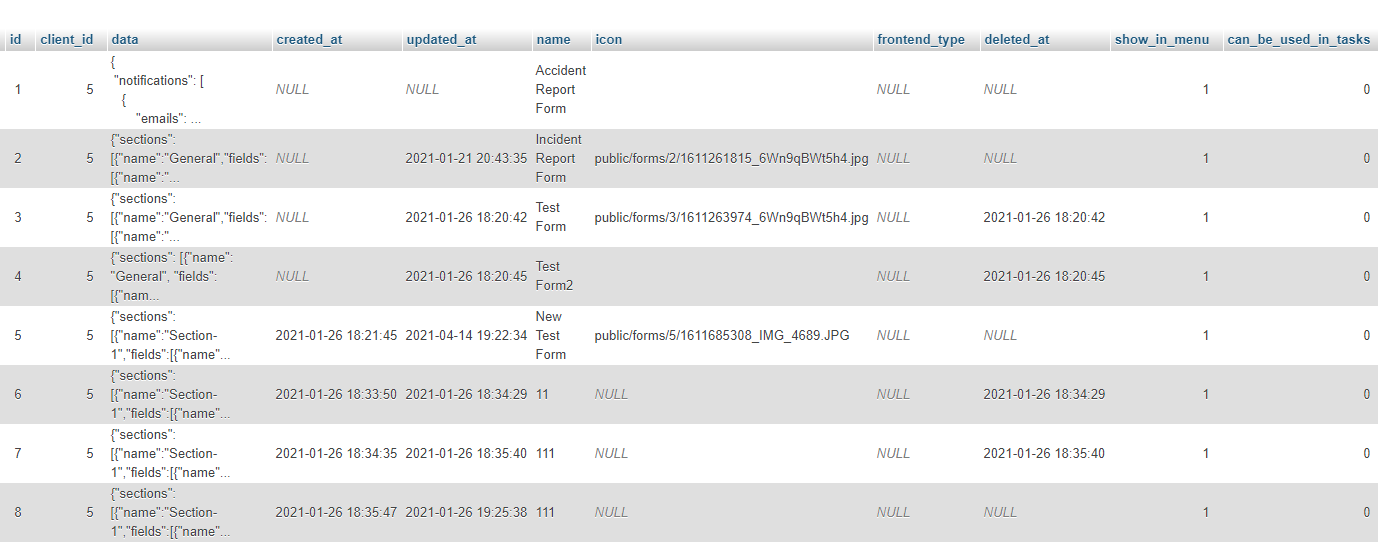


Рис. 3.6. Таблиця бази данних – forms.

# 3.1.4 Маршрутизація

Заглушки маршрутів

Тепер додамо кілька маршрутів в наш додаток. Маршрути використовуються для зв'язку URL з контролерами або анонімними функціями, які повинні бути виконані, коли користувач переходить на дану сторінку. За замовчуванням всі маршрути Laravel визначені в файлі  **routes/web.php**, який автоматично додається в кожен новий проект.

Для нашого застосування будуть потрібні такі маршрути:

* маршрут для виведення на екран списку форм,
* маршрут для додавання нових форм
* маршрут для редагування існуючих форм
* маршрут для видалення існуючих форм.
* маршрут для завантаження іконок форм
* маршрут для вибору Autocomplete у поляї форм
* маршрут для вибору людей яким буде доступна секція в формі

Напишемо заглушки для всіх цих маршрутів в файлі **routes/web.php**:

Route::get('/form-builder', 'HomeController@formBuilder')->name('form-builder');  
Route::post('/form-builder/', "FormBuilder@create")->name('form-create');  
Route::put('/form-builder/{form?}', "FormBuilder@update")->name('form-set');  
Route::get('/form-builder/{form?}', "FormBuilder@view")->name('form-get');  
Route::delete('/form-builder/{form?}', 'FormBuilder@delete')->name('form-remove');  
Route::get('/vocabulary/{user?}', 'VocabularyController@list')->name('vocabulary.list');  
Route::get('/vocabulary/single/{vocabulary?}', 'VocabularyController@getSingleVocabulary')->name('vocabulary.single');  
Route::get('/form-builder/list/{user?}', 'FormBuilder@list')->name('forms.list');  
Route::post('/form-builder/upload/', 'FormController@ajaxImageUploadPost')->name('forms.upload');  
Route::get('/roles/{user?}', 'RoleController@list')->name('roles.list');

Рис. 3.7. Вміст файлу routes/web.php

**Вивод уявлення**

Зповнимо маршрут /form-builder. За цим маршрутом намалюемо HTML-шаблон, який містить список форм.

У Laravel все HTML-шаблони зберігаються в каталозі **resources / views**, і ми використовувуемо допоміжну функцію view (), щоб повернути один з цих шаблонів по нашому маршруту:

Route::get('/form-builder', 'HomeController@formBuilder')->name('form-builder');

Рис. 3.8. Визов шаблону form-builder

Передача forms в функцію view () створить екземпляр об'єкта View, який відповідає шаблоном:  
resources/views/form-builder/form-builder.blade.php.

**Визначаємо макет**

Майже всі веб-додатки використовують один макет на всіх своїх сторінках. Наприклад, у нашого застосування є верхня панель навігації, яка була присутня б на кожній сторінці (якби у нас їх було більше однієї). Laravel спрощує використання цих загальних функцій на всіх сторінках, використовуючи макети Blade.

Всі уявлення Laravel зберігаються в resources/views. Визначимо представлення нового макета в resources/views/layouts/app.blade.php. Розширення .blade.php дає фреймворку команду використовувати механізм шаблонної обробки Blade, щоб отрисовать це уявлення. Звичайно, в Laravel ви можете використовувати і прості PHP-шаблони. Однак Blade дозволяє швидко написати простий і невеликий шаблон.

Наше уявлення app.blade.php має вигляд:

Код знаходиться у додатку 2

**@yield ('content')** в макеті - це спеціальна Blade-директива для вказівки всім дочірнім сторінкам, спадщини цей шаблон, де вони можуть впровадити свій вміст. Давайте визначимо дочірнє уявлення, яке буде використовувати цей макет і виводити його основний контент.

**@can ('build', \ App \ Facades \ Auth :: user ())** в макеті - це спеціальна Blade-директива для приховування блоків для користувачів у яких немає доступу до контенту або показу блоку користувачам які авторизовані і мають доступ.

# 3.2.1 Установка Vue.js

Найшвидший і найправильніший спосіб установки vue.js через Vue CLI.

**Vue CLI.**

Для того щоб повноцінно використовувати vue.js в своїх проектах, як правило одного vue.js мало, вам швидше за все будуть потрібні доп. плагіни до vue, такі як vuex.js, плагіни для підтримки застарілих браузерів і інші залежності.

Щоб не витрачати на це багато часу, є можливість почати працювати з vue написавши один рядок в консолі, в цьому нам і допоможе Vue CLI. Щоб розгорнути стартовий набір, достатньо написати однин рядок:

vue create project-name

**Створення проекту**

1. **установка node.js**   
   npm install -g @ vue / cli
2. **Установка Vue CLI.** Відкриваємо консоль і запускаємо інсталятор npm install -g @ vue / cli
3. **Створення проекту на vue.js** В консолі переходимо в ту директорію, в якій ви хочете розгорнути проект і запускаємо команду:   
   vue create FormBuilder

У файлі resources/js/app.js ініціюемо наш однофайловий Vue компонент:

var Vue = require('vue');  
  
require('./bootstrap');  
  
***window***.Vue = require('vue');  
Vue.mixin(require('./mixins/asset.js'));  
  
const files = require.context('./', true, /\.vue$/i)  
files.keys().map(key => Vue.component(key.split('/').pop().split('.')[0], files(key).default))  
Vue.component('form-builder', require('./components/FormBuilder.vue').default);  
  
***window***.onload = function () {  
 var app = new Vue({el: '#form-builder'});  
}

Рис. 3.9. Компонент Vue

Для того щоб зібрати Vue додаток, треба у консолі написади наступний рядок:   
npm run build

# 3.2.2 Написання Vue додатку

Создамо діректорію resources/js/components.  
У цій діректорії будуть знаходитись всі наші Vue домпоненти. Для початку зробимо Головний компонент в якому буде знаходитись список усіх доступних форм.

Создамо компонент: resources/js/components/FormSelectionPage.vue

Код знаходиться у додатку 3

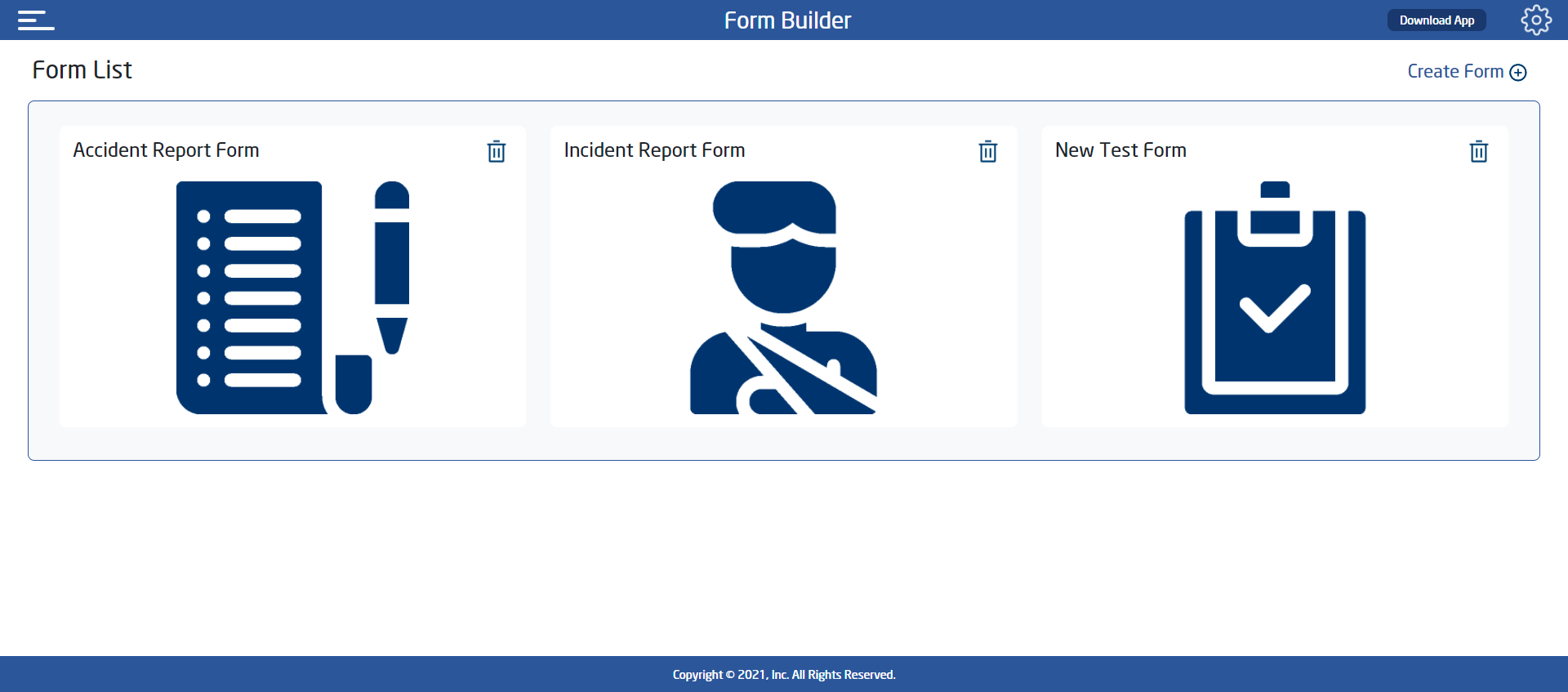


Рис. 3.10. Сторінка зі списком форм

У цьому компоненті знаходяться “картки” існуючих форм з назвою, іконкою форми та з кнопкою видалення форми. Також у правій верхній частині знаходиться кнопка для створення нової форми у Конструкторі форм.

Далі розробимо додаток самого **Конструктора Форм**, для цього потрібно реалізувати Html Drag 'n' Drop.

# 3.2.3 Html Drag 'n' Drop

Інтерфейс HTML Drag and Drop дозволяє додаткам використовувати функціонал перетягування в браузерах. Користувач може вибрати перетягуються елементи мишею, перетягнути до елементу скидання і кинути їх, відпустивши кнопку миші. Під час перенесення за покажчиком слід напівпрозорий образ перетягуваного елемента.

При створенні веб-сайтів, розширень і XUL-додатків можна налаштовувати, які саме елементи можна перетягувати, яку зворотний зв'язок вони при цьому будуть створювати, і які елементи будуть створювати область скидання.

Реалізувати Drag’n’Drop будемо з допомогою плагіна **Vue.**Draggable**.**

Плагін знаходиться у відкритому доступі за посиланням: https://github.com/SortableJS/Vue.Draggable

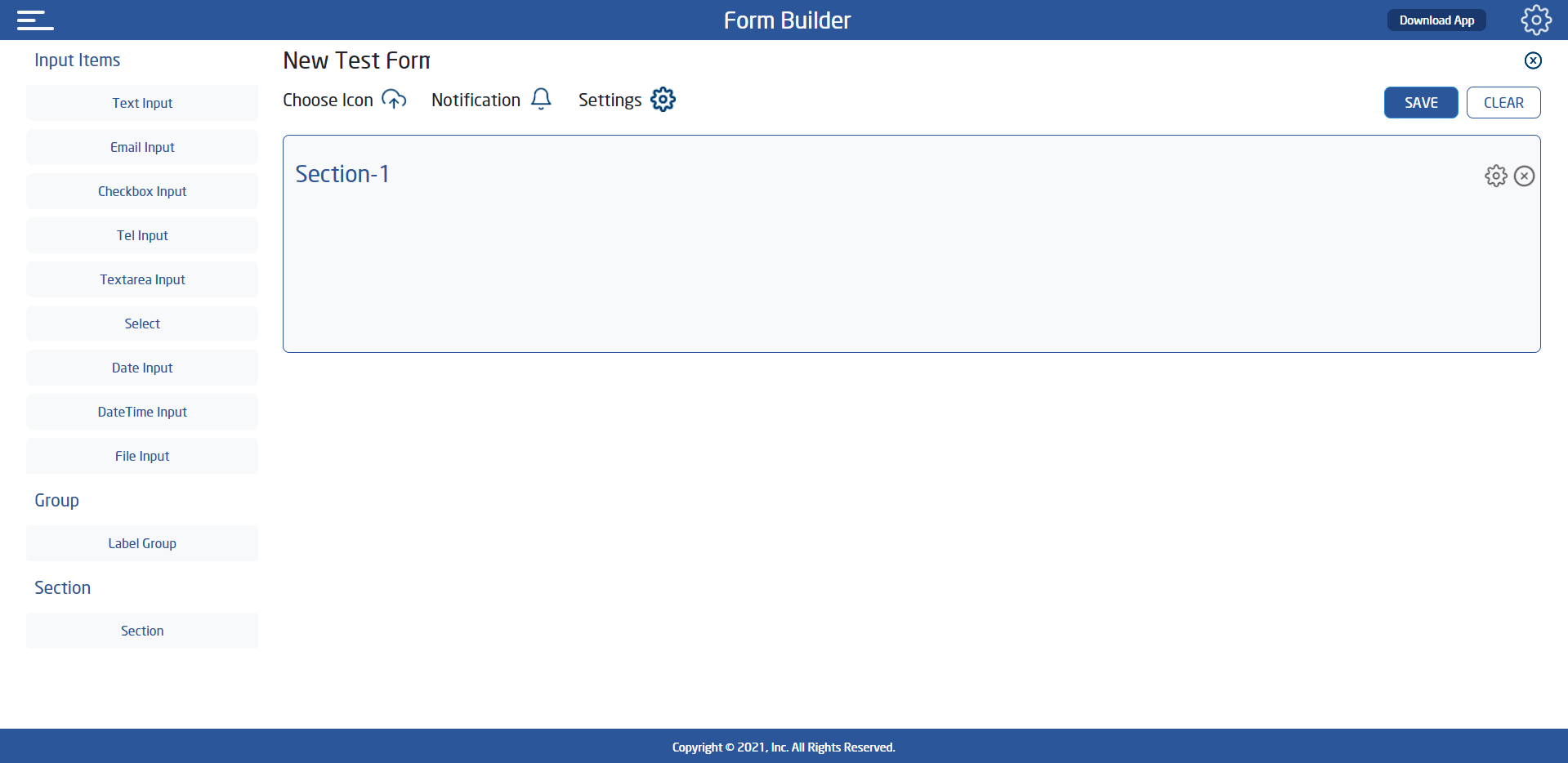


Рис. 3.11. Сторінка Конструктора Форм

# 3.2.4 Сторінка Конструктора Форм

Код знаходиться у додатку 4.

Зліва знаходииться бокова панель, в ній розташовані Елементи які можливо використовувати для створення форми.

Зверху у правій та основній частині, знаходяться:

* поле введення імені форми

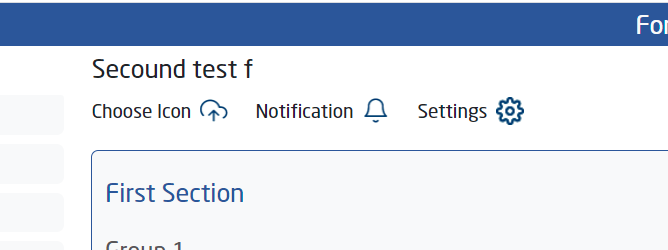


Рис. 3.12. Поле введення імені форми

* панель для вибору іконки форми

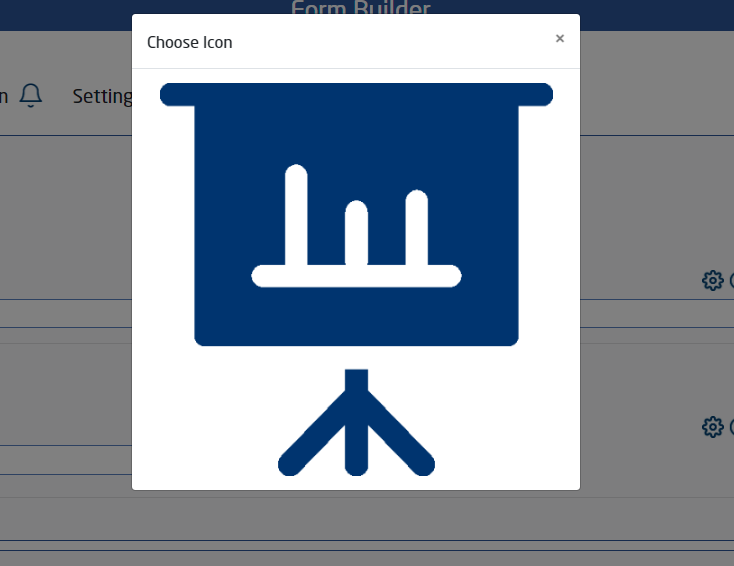


Рис. 3.13. панель для вибору іконки форми

* створення оповіщення по формі

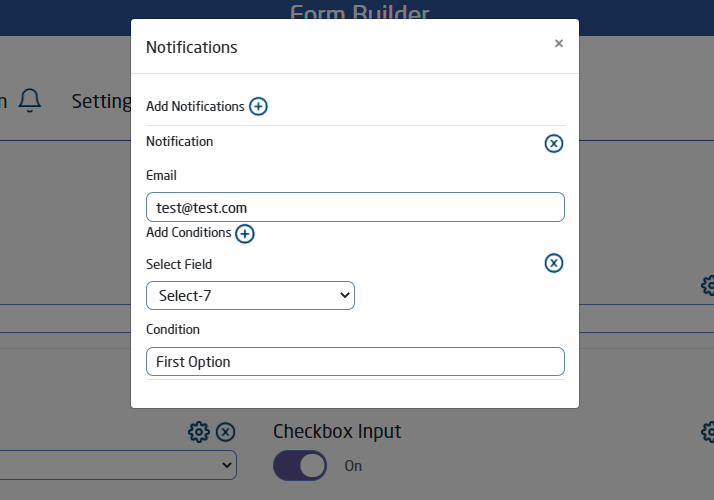


Рис. 3.14. Створення оповіщення по формі

* кнопка налаштувань форми

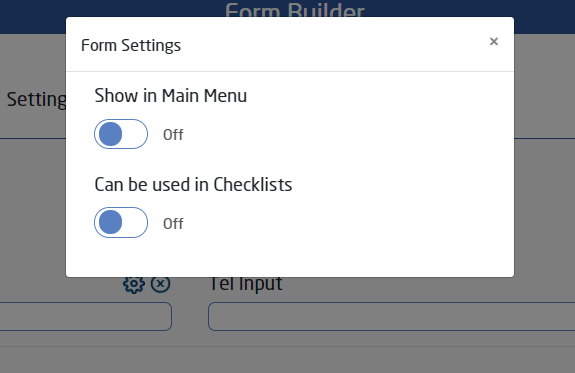


Рис. 3.15. Налаштування форми

* кнопки збереження та очищення форми.

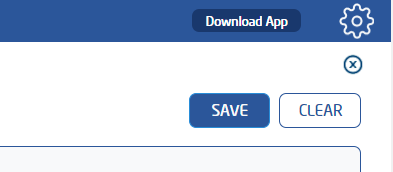


Рис. 3.16. Кнопки збереження та очищення форми

Нижче знаходиться поле для формування форм.

Формувати форму можливо завдяки основним типам полей для введення данних які є у HTML5, а саме:

* input type text,
* input type email,
* checkbox,
* textarea,
* select,
* date та date and time input,
* file input

Також для для простоти і легкості розуміння до чого відносяться певні поля введення, було розроблене поле Label Group. У це поле можливо вложено розмістити інши поля як показано на малюнку знизу.

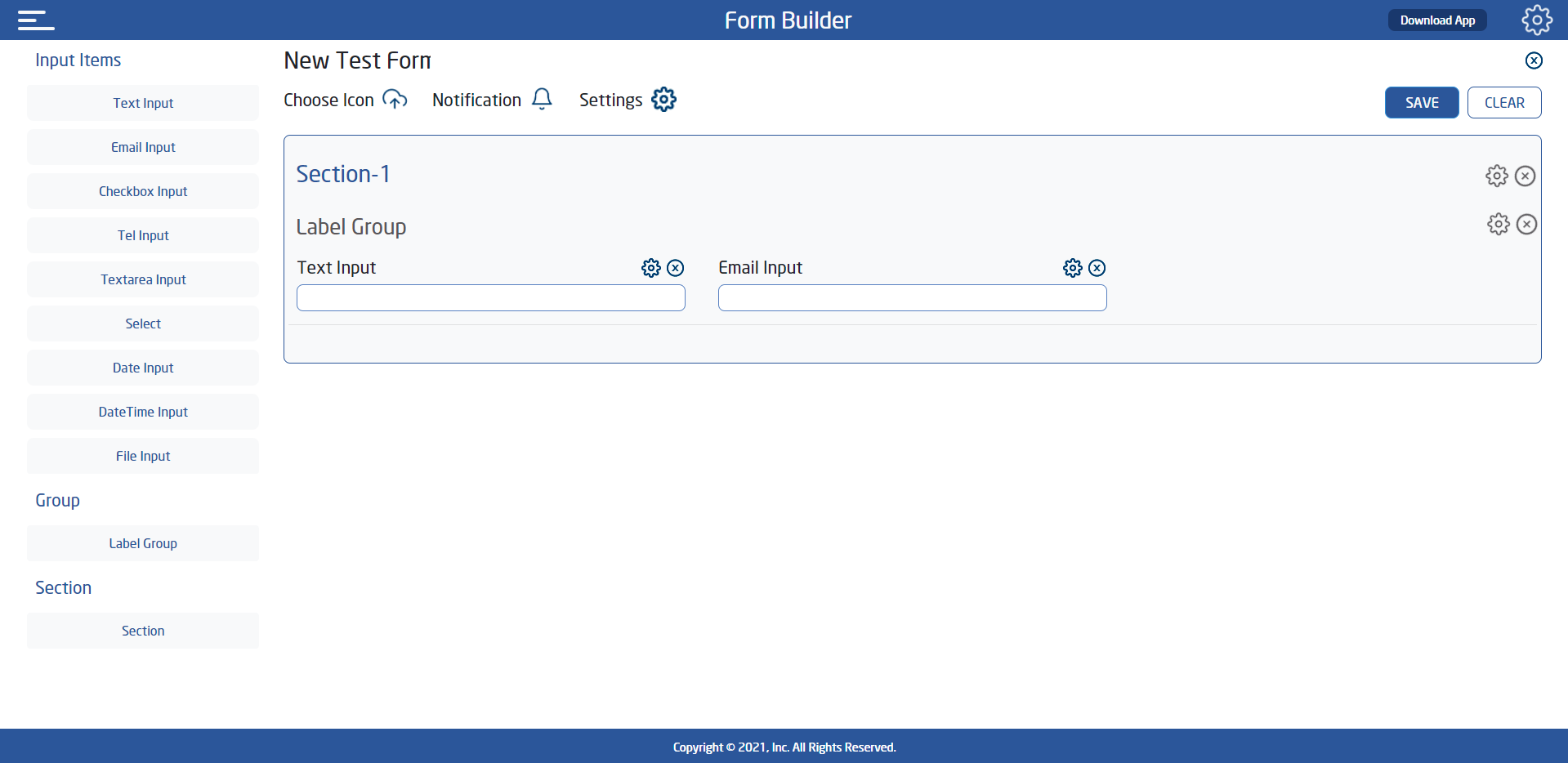


Рис. 3.17. Демонстрація поля Label Group

Поле Section стоврене для того, шоб можливо було робити багатостронічні форми.

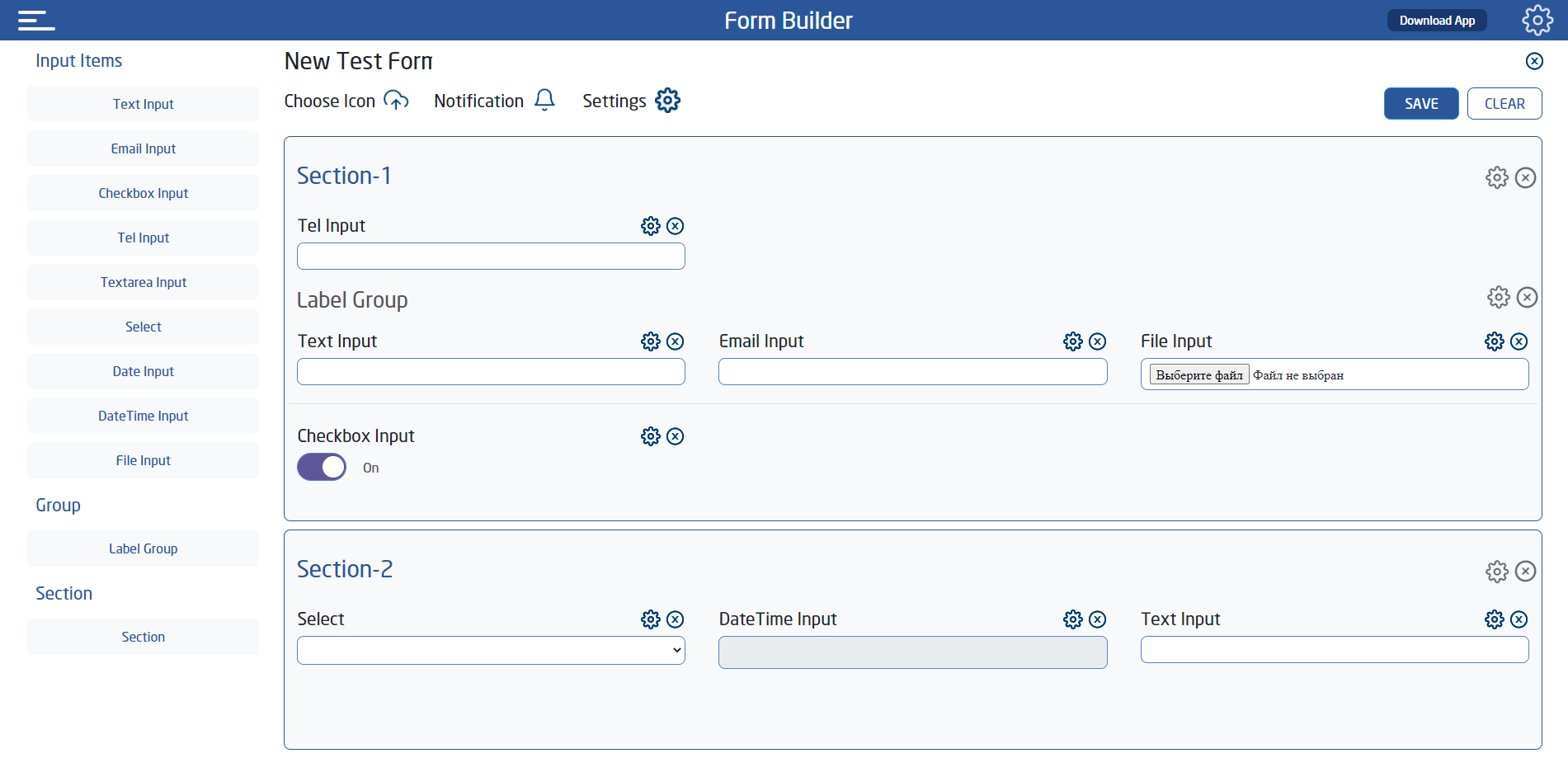


Рис. 3.18. Демонстрація поля Section

У кожне поле також були добавлені свої, індивідуальні, налаштування.

**Поле налаштуваня для Text Input.**

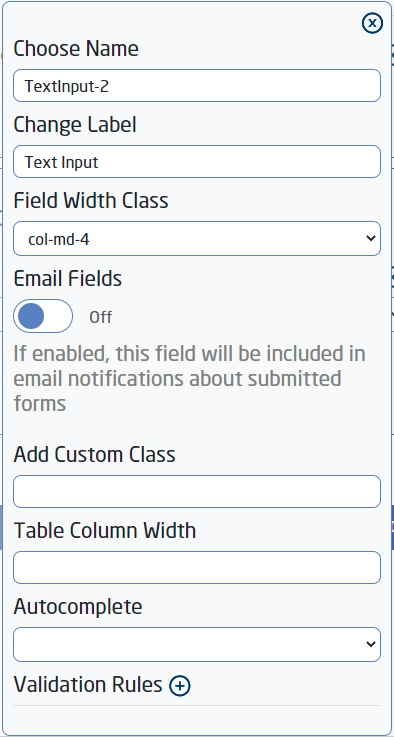


Рис. 3.19. Налаштування поля Text input

**Choose Name** – це поле в якому можно задати унікальне і’мя для компонента. За замовчуванням Генеруеться автоматично.

**Change Label** – у цьому полі змінюеться Lable поля.

**Field Width Class** – тут можно вибрати 1 из 3 варіантів ширини поля 1/3, 1/2 чи 1/1 з домопомою додавання класу.

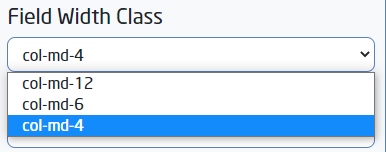


Рис. 3.20. Параметр Field Width Class

**Email Fields** – якщо активно, то це поле буде включено в повідомлення по електронній пошті про підправлені форми.

**Add Custom Class** – через це поле, можливо добавити спецыальний класс.

**Table Column Width** – у цьому полі можливо задати ширину столбця под це поле, на сторінкі зі звітами о подіях.

**Autocomplete** – якщо є, то можна зробити цє поле Autocomplete та вибрати я закого списку його брати.

**Validation Rules** – дає можливість добавляти правила валідації для форми. Доступні при основних правила, обязковість поля та максимальна і мініпальна кількість символів.

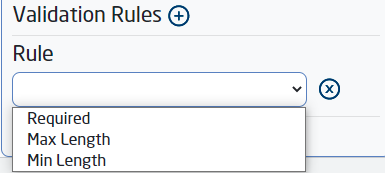


Рис. 3.21. Параметр Validation Rules

**Поля налаштуваня для Email Input.**

Поля налаштування для Email Input такі ж самі як и для Text Input.

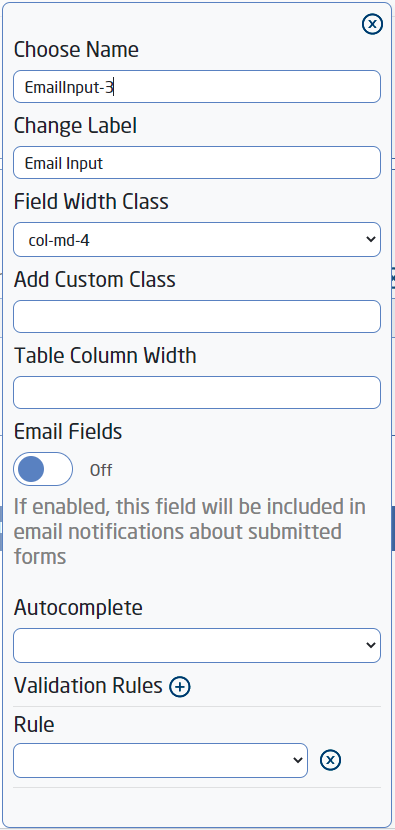


Рис. 3.22. Налаштування поля Email input

**Поля налаштування для Checkbox Input**

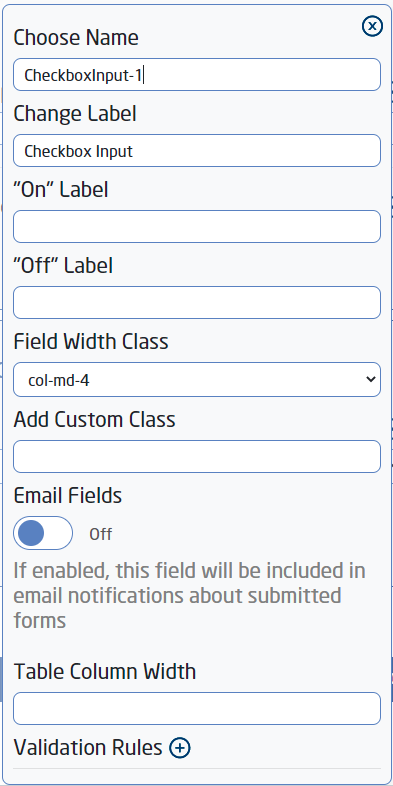


Рис. 3.23. Налаштування поля Checkbox input

Поля налаштування у Checkbox Input мойже такі самі як і у Text Input, окрім того , що у Checkbox Input немае поля налаштування Autocomplete та добавлені два поля “On” Label та “Off” Label.

“On” Label та “Off” Label – дає змогу міняти Label при активному чи

неактивному Checkbox Input. За замовчуванням “On” та” Off” відповідно.

**Поля налаштування для Tel Input.**

Поля налаштування для Tel Input такі ж самі як и для Text Input**.**

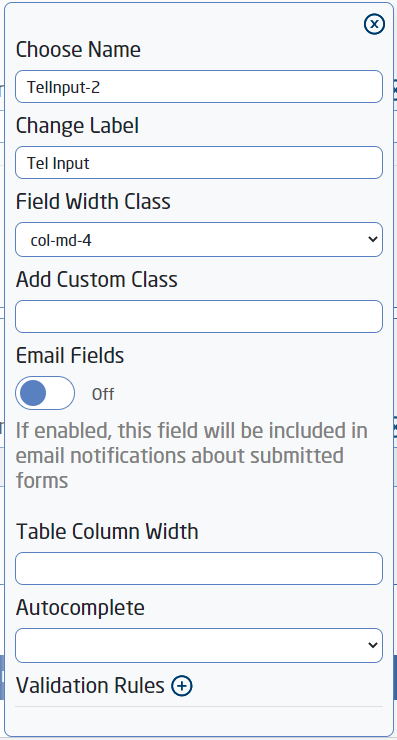
****

Рис. 3.24. Налаштування поля Tel input

**Поля налаштування для Textarea.**

Поля налаштування для Textarea майже такі ж самі як и для Text Input, немає тільки поля Autocomplete.

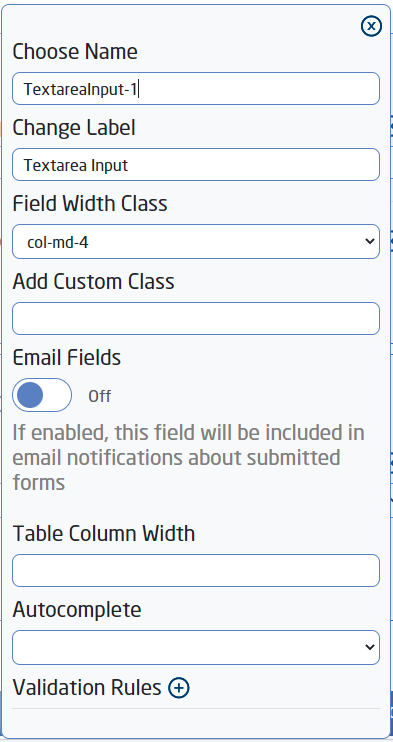
****

Рис. 3.25. Налаштування поля Textarea

**Поля налаштування для Select.**

Поля налаштування для Textarea майже такі ж самі як и для Text Input.

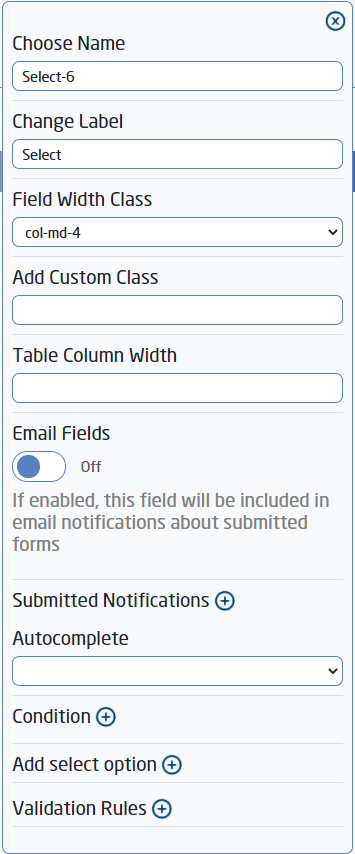
****

Рис. 3.26. Налаштування поля Select

Добавлені такі поля:

**Submitted Notifications** – у цьому полі можливо вибрати сповіщення, час сповіщення, значення умови сповіщення та оператор стану сповіщення.

****

Рис. 3.27. Параметр Submitted Notifications

**Conditions –** дає можливість, зробити залежність ативності поля від значення іншого.

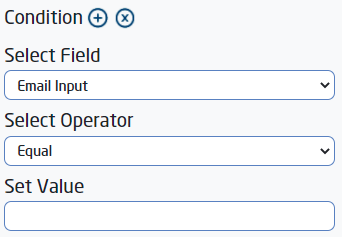


Рис. 3.28. Параметр Conditions

**Add select option –** у цьому полі можно добавити до Select опції для вибору.

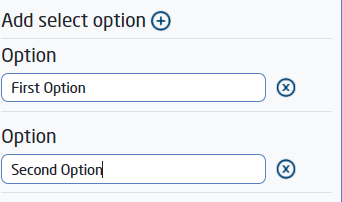


Рис. 3.29. Параметр Add select option

**Поля налаштування для Date Input та DateTime Input.**

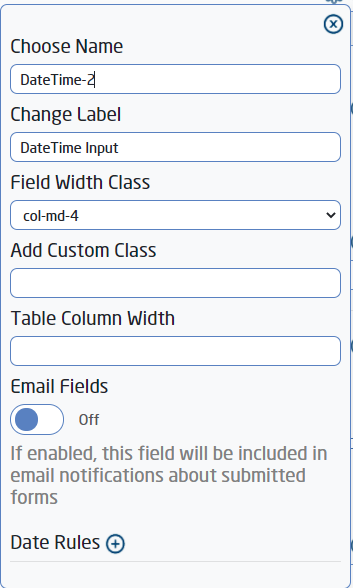


Рис. 3.30. Налаштування полей Date Input та DateTime Input

Поля налаштування для Date Input та DateTime Input майже такі ж самі як и для Text Input. Видалено поле Autocomplete та поле Validation замінено на Date Rules.

Date Rules – дає змогу добавляти правила для ввода Дати.

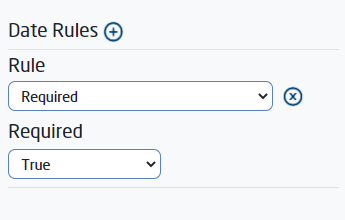


Рис. 3.31. Параметр Date Rules

**Поля налаштування для File Input.**

Поля налаштування для File Input майже такі ж самі як и для Text Input.

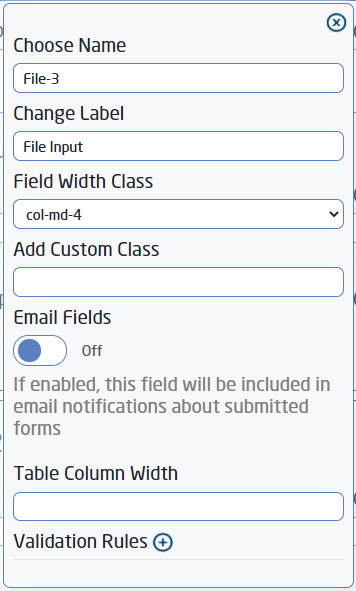


Рис. 3.32. Налаштування поля File input

Видалено поле Autocomplete.

**Поля налаштування для Label Group.**

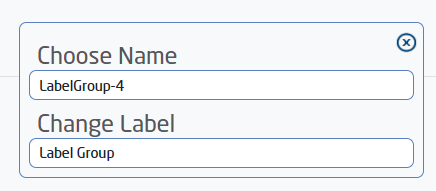
****

Рис. 3.33. Налаштування поля Label Group

Label Group має всього два поля для налаштування:

**Choose Name** – у цому полі можно задати унікальне і’мя для компонента. За замовчуванням Генеруеться автоматично.

**Change Label** – цє поле дає можливість змінювати Lable поля.

**Поля налаштування для Section.**

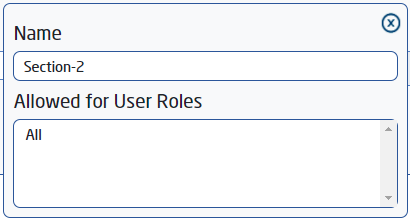
****

Рис. 3.34. Налаштування поля Section

Label Group має всього два поля для налаштування:

**Name** - це поле в якому можно задати унікальне і’мя для компонента. За замовчуванням Генеруеться автоматично.

**Allowed for User Role** – цє поле дає змогу контролювати хто зможе бачити цю секцію, а для кого вона буде схована.

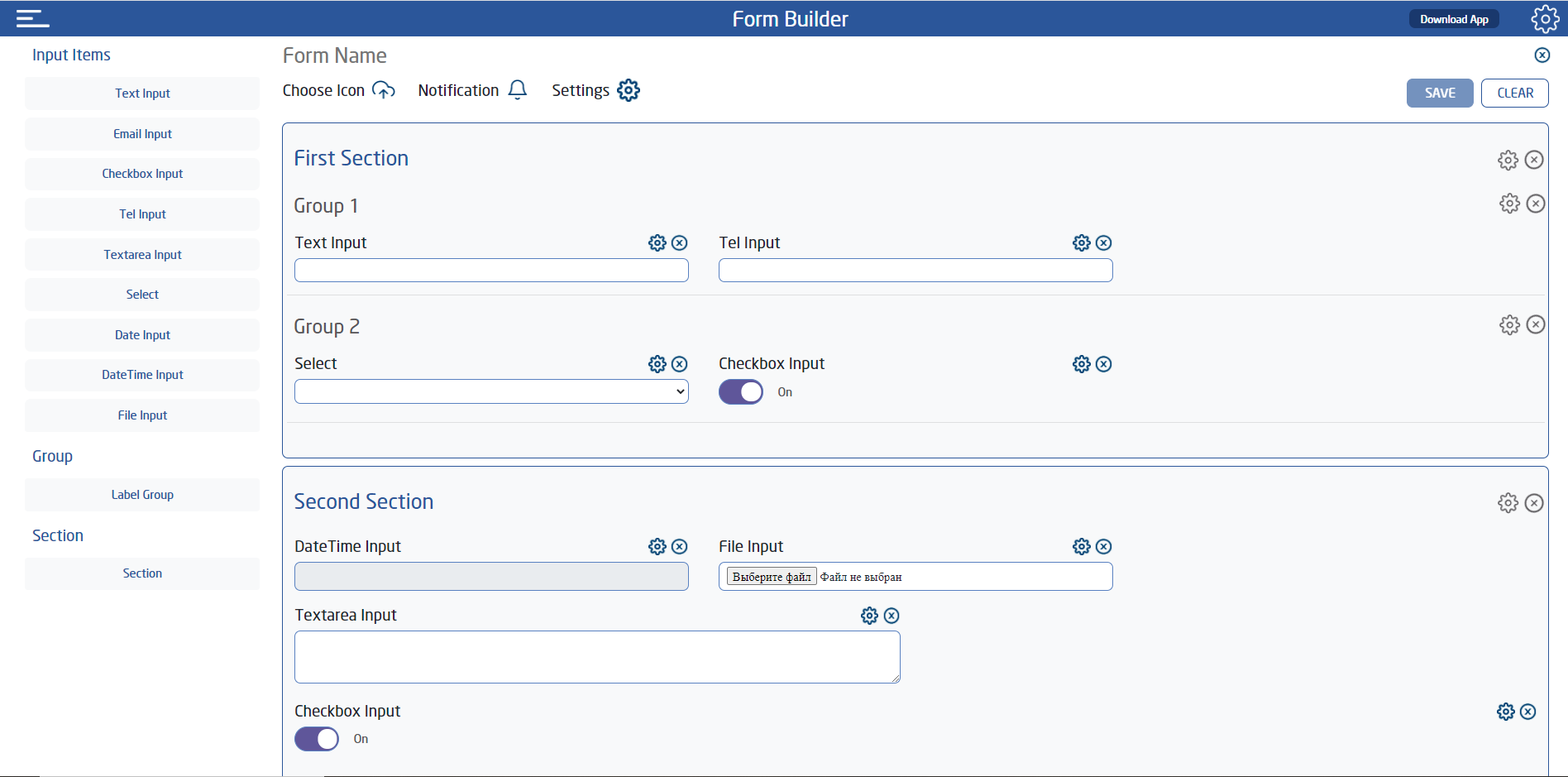
****

Рис. 3.35. Приклад форми

**Перегляд та заповнення форми.**

Якщо у налаштуванні форми зробити автивним поле Show in Main Menu

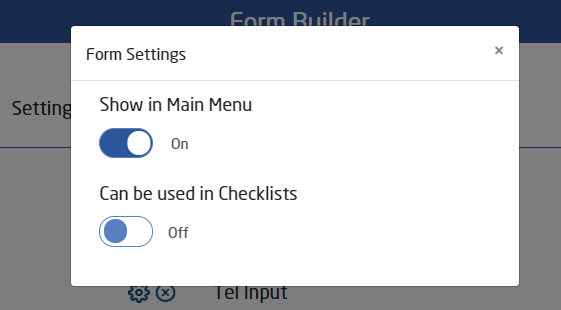


Рис. 3.36. Налаштування форми

То форма відобразиться на головній сторіці



Рис. 3.37. Головна сторінка

Після цого можно використовувати форму, щоб на неї перейти треба зробики клік на іконку або назву форми.

Код читача форм знаходиться у додатку 5.

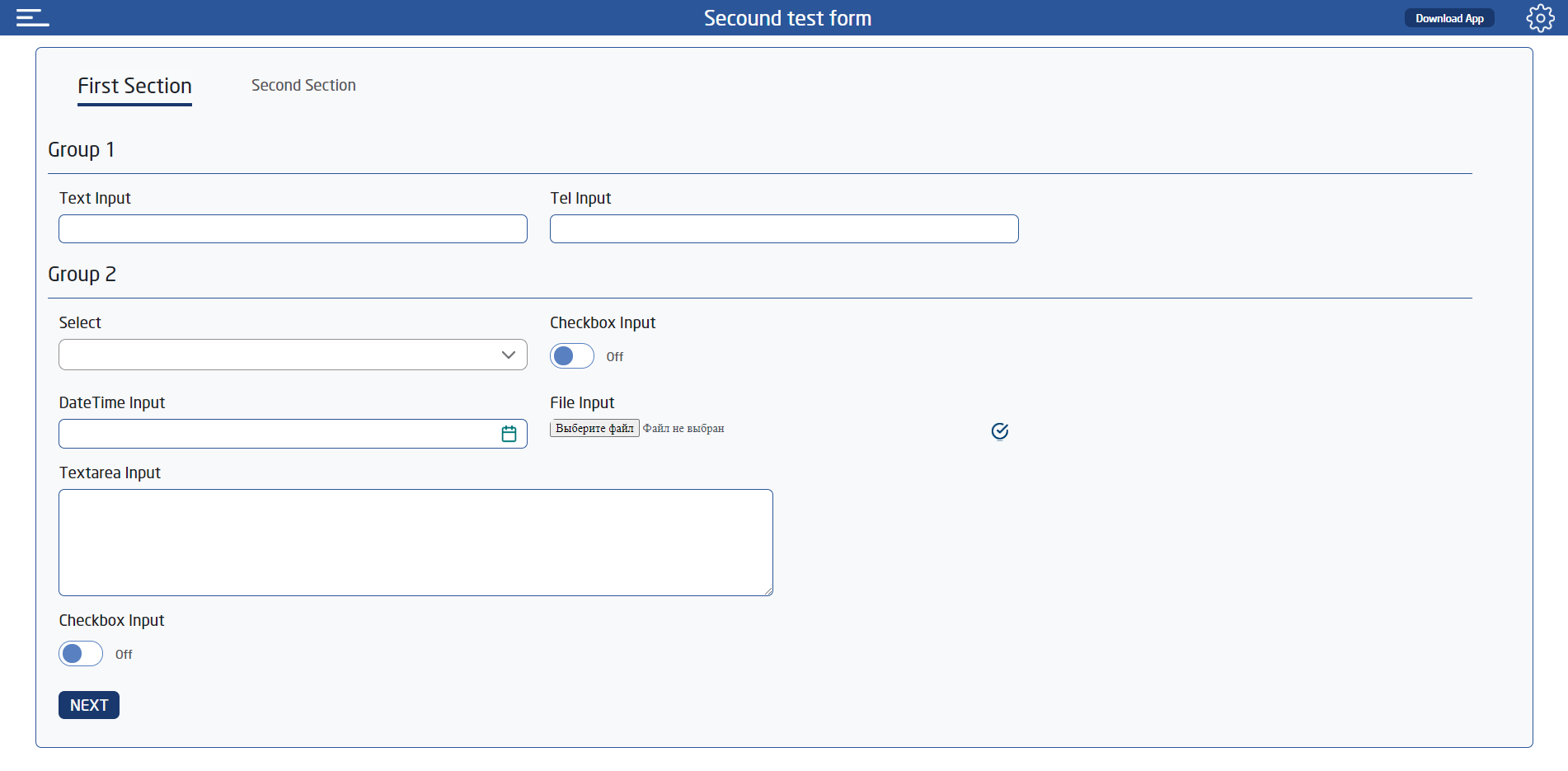


Рис. 3.38. Перший таб створеної форми

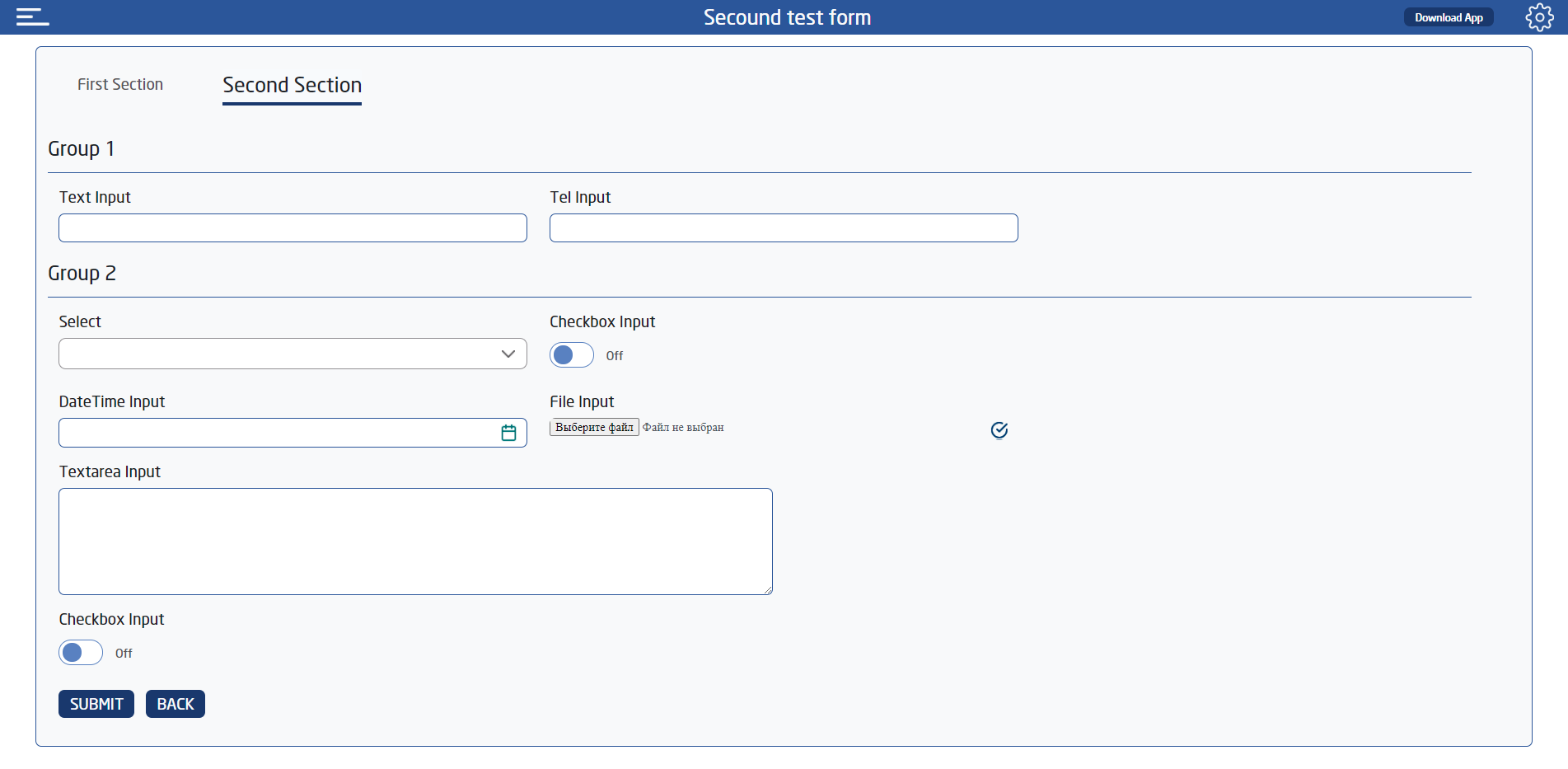


Рис. 3.38. Другий таб створеної форми

**Висновок**

Був розроблений корпоративний Веб-додаток для фіксації та оповіщення робітників о надзвичайних подіях. У ході роботи були визначені js та php фреймворки які напйкраще підходили під розробку додатку. Такими фремворками стали Vue.js та Laravel php.

Vue.js – став кращим вибором через те, що він чудово підходить для роробки односорінкового компоненту, дуже легкий фреймворк (з точки зору швидкості) та його дуже леко інтегрувати в php фреймворк laravel.

Laravel – був обраний через те, що цє найпотужніший фреймворк з великою кількість готових шаблонів та компонентів та дуже легко інтегрується з vue.js.

В розробці додатку головною частиною була розробка конструктора форм, в якому можливо було конструювати форми, під свої потреби. Також була розроблена система оповіщення на основі заповнених даних у формі. Система оповіщення може працювати як оповіщення електронную поштою так і оповіщення на смартфон у виді SMS.

# 4. Список літератури, що використовувалась:

1. Веб-технології для розробників: Веб-сайт.

URL: https://developer.mozilla.org/ (дата зверненя: 02.06.2021) .

1. Vue.js – прогресивний JavaScript-фреймворк:   
   Веб-сайт. URL: https://vuejs.org (дата зверненя 02.06.2021)
2. Angular.js – сучасна платформа веб розробників.  
    Веб-сайт. URL: https://angular.io (дата зверненя 02.06.2021)
3. React - JavaScript-бібліотека для создания корістувацькіх інтерфейсів. Веб-сайт. URL: https://reactjs.org (дата зверненя 02.06.2021)
4. Laravel - php-фреймворк нового покоління. Документація версії 8.х.   
   Веб-сайт. URL: https://laravel.su/docs/8.x/ (дата зверненя 13.06.2021)
5. Хабр. Symfony – актуальність. Веб-сайт. URL: https://habr.com/ru/company/otus/blog/344726/ (дата зверненя 13.06.2021)
6. Хабр. Особливості npm і зберігання node\_modules в git.   
   Веб-сайт. URL: https://habr.com/ru/post/185200/ (дата зверненя 13.06.2021)
7. GitHub. Vue.draggable. Веб-сайт. URL: https://github.com/SortableJS/Vue.Draggable (дата зверненя 13.06.2021)

# Додаток 1

/\*\*  
 \* @param Client|integer $client  
 \* @return Form[]  
 \*/  
 public static function getMenuFormsForClient($client)  
 {  
 $clientId = $client;  
 if ($clientId instanceof *Client*) {  
 $clientId = $clientId->getKey();  
 }  
 return self::where(['client\_id' => $clientId, 'show\_in\_menu' => *1*])->get();  
 }  
  
 public static function getTaskFormsForClient($client)  
 {  
 $clientId = $client;  
 if ($clientId instanceof *Client*) {  
 $clientId = $clientId->getKey();  
 }  
 return self::where(['client\_id' => $clientId, 'can\_be\_used\_in\_tasks' => *1*])->get();  
 }  
}

# Додаток 2

<a href="{{ route('form', ['locale'=> app()->getLocale(), 'id' => $key]) }}"><li class="es-items"><span class="es-icon-form" style="background: url({{ url($form['icon']) }})"></span>{{ $form['name'] }}</li></a>  
 @endforeach  
 <li class="es-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" id="navbarReports" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
 <span class="es-reports-icon"></span>Reports  
 </a>  
 <div class="dropdown-menu es-dropdown-menu" aria-labelledby="navbarReports">  
 @foreach ( $allowedForms as $key => $form)  
 <a class="dropdown-item" href="{{ route('history-form', ['locale'=> app()->getLocale(),'form'=>$key]) }}">  
 <div class="es-items">  
 <span class="es-icon-form" style="background: url({{ url($form['icon']) }})"></span>  
 {{ $form['name'] }}  
 </div>  
 </a>  
 @endforeach  
 @foreach ( $supervisedForms as $key => $form)  
 <a class="dropdown-item" href="{{ route('history-form', ['locale'=> app()->getLocale(),'form'=>$key]) }}">  
 <div class="es-items">  
 <span class="es-icon-form" style="background: url({{ url($form['icon']) }})"></span>  
 {{ $form['name'] }}  
 </div>  
 </a>  
 @endforeach  
 </div>  
 </li>  
 <li class="es-item dropdown">  
 <a class="nav-link dropdown-toggle" href="#" id="navbarDashboard" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
 <span class="es-dashboard-icon"></span>Dashboard  
 </a>  
 <div class="dropdown-menu es-dropdown-menu" aria-labelledby="navbarDashboard">  
 @foreach ( $allowedForms as $key => $form)  
 <a class="dropdown-item" href="{{ route('dashboard', ['locale'=> app()->getLocale(), 'formId' => $key]) }}">  
 <div class="es-items">  
 <span class="es-icon-form" style="background: url({{ url($form['icon']) }})"></span>  
 {{ $form['name'] }}

# Додаток 3

if ($('#app').hasClass('admin')) {  
 that.clientId = response.data.data.client\_id.toString();  
 // that.clientMemberId = response.data.data.user\_id.toString();  
 }  
  
 that.formToEdit = response.data;  
 that.formName = response.data.data.name;  
 })  
 .catch((error) => {  
 ***console***.log(error);  
 })  
 },  
 deleteForm(id) {  
 var confirmRemove = confirm("Do you really want to delete form?");  
 var that = this;  
  
 if (confirmRemove) {  
 axios.delete(that.form\_builder\_delete\_url + '/' + id)  
 .then((response) => {  
 that.loadForms()  
 })  
 .catch((error) => {  
 ***console***.log(error);  
 })  
 }  
 },  
 loadForms() {  
 var that = this;  
 that.isLoad = true;  
 axios.get(that.form\_builder\_list\_url + '/' + that.client\_member\_id)  
 .then((response) => {  
 that.forms = response.data;  
 if (response.data[0]) {  
 that.isShowFormList = true;  
 that.onloadHash();  
 } else {  
 that.isShowFormList = false;  
 }  
 that.isLoad = false;  
 })  
 .catch((error) => {  
 that.showError = true;  
 that.errorMessage = error;  
 that.isLoad = false;  
 })  
  
 ***console***.log('form\_image\_upload\_url', that.form\_image\_upload\_url);  
 },  
 getBaseURL() {  
 return ***window***.APP\_URL;  
 },  
 hideError() {  
 this.showError = false;  
 },  
 onloadHash() {  
 if (***window***.location.hash) {  
 var formId = ***window***.location.hash.replace('#form=', '');  
 ***window***.location.hash = '';  
 this.editForm(formId);  
 }  
 }

# 

# Додаток 4

<template>  
 <div class="form-builder-wrapper container">  
 <errorComponent  
 v-if="showError"  
 :message="errorMessage"  
 v-on:show="hideError"  
 >  
 </errorComponent>  
 <div class="row">  
 <div v-if="screenWidth > 990" class="col-2">  
 <div class="wrapper">  
 <h3 class="collapse-btn" data-toggle="collapse" data-target="#collapseInputs" aria-expanded="true" aria-controls="collapseInputs">Input Items</h3>  
 <div id="collapseInputs" class="collapse show" aria-labelledby="headingInputs" >  
 <draggable  
 class="dragArea list-group"  
 :list="toolList"  
 :group="{ name: 'builder-group-inputs', pull: 'clone', put: false }"  
 :clone="cloneObj"  
 @change="log"  
 >  
 <div class="list-group-item inputs w-100" v-for="input in toolList" :key="input.name">  
 {{input.label}}  
 </div>  
 </draggable>  
 </div>  
 <h3 class="collapse-btn"  
 data-toggle="collapse"  
 data-target="#collapseGroup"  
 aria-expanded="true"  
 aria-controls="collapseInputs">Group</h3>

# Додаток 5

<?*php*/\*\* @var \App\Form\HtmlForm $form \*/  
?>  
<*form id*="validateForm" *action*="{{ route('form-submit') }}" *method*="post" *enctype*="multipart/form-data" \>  
 <*input type*="hidden" *name*="form-id" *value*="{{ $form->id }}">  
 <*input type*="hidden" *name*="form-is-edit" *value*="{{ old('is\_edit') ?? 0 }}">  
 <*input type*="hidden" *name*="submitted-form-id" *value*="{{ old('submitted\_form\_id') }}">  
 @foreach( $form->hiddenFormFields as $name => $value )  
 <*input type*="hidden" *name*="{{$name}}" *value*="{{$value}}">  
 @endforeach  
 <*div* class="top-wrapper">  
 @*csrf* <*ul* class="nav nav-tabs">  
 @foreach( $form->sections as $key => $section )  
 <?*php* if($section->disabled)  
 continue;  
 ?>  
 <*li* class="nav-item" *data*-*index*="{{ $key }}">  
 <*a* class="nav-link @if($key == 0) active @endif" *data*-*toggle*="tab" *href*="#section-{{ $section->id }}">{{ $section->name }}</*a*>  
 </*li*>  
 @endforeach  
 </*ul*>  
 {{--<*div* class="icon"><*img src*="{{url("../*images*/*Group 12.png*")}}"></*div*>--}}  
 </*div*>  
 <*div* class="components-wrapper">  
 <*div* class="tab-content pt-3">  
 @foreach( $form->sections as $key => $section )  
 <?*php* if($section->disabled)  
 continue;  
 ?>  
 <*div* class="tab-pane fade @if($key == 0) show active @endif" *id*="section-{{ $section->id }}">  
 <*div* class="form-inner-wrapper">  
 @*php* echo $section;  
 @*endphp* </*div*>  
 </*div*>  
 @endforeach  
 </*div*>  
 </*div*>  
 <*button* class="save-incomplete" *type*="submit" *name*="save" *value*="incomplete">{{ \_\_('Submit') }}</*button*>  
{{-- <*button* class="save-complete" *type*="submit" *name*="save" *value*="complete">{{ \_\_('Save & Mark as Completed') }}</*button*>--}}  
 <*button type*="button" *id*="next-tab">*Next*</*button*>  
 <*button type*="button" *id*="prev-tab">*Back*</*button*>  
</*form*>  
<*script src*="{{ asset('js/validator.js') }}" ></*script*>