ЗМІСТ

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

MVC - Model-View-Controller

BSD - Berkeley Software Distribution

CDA - Content Delivery Application

CMA - Content Management Application

CMS - Content Management System

API - Application Programming Interface

CSS - Cascading Style Sheets

DVB - Digital Video Broadcasting

FTP - File Transfer Protocol

GPL - General Public License

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

ISN - Initial Sequence Number (початковий порядковий номер)

ВСТУП

Мета - поширення волонтерського руху в Україні. Велика ідея - утвердження волонтерства для кожного. Можливість допомогти українцям брати участь у волонтерських проектах.

З досвіду різних проектів можна чітко побачити, наскільки корисним для усіх може бути волонтерство - і для підлітків, і для дітей, і для професіоналів. Тому, з одного боку, з’явиться можливість будувати середовище, де люди можуть себе реалізувати у культурних і освітніх проектах, з іншого - підтримувати волонтерів та створювати для них різні можливості. Вберегти людей від вигорання і навчити бути максимально ефективними у своїй справі.Хочеться, щоб в Україні було набагато більше волонтерів. Тому мета - мобілізувати і допомогти залучити до волонтерського руху якомога більше людей.

Працювати слід у чотирьох основних напрямках.

Перший - формування волонтерських спільнот у найбільших містах. В основі формування спільнот - об’єднання великої кількості людей, які поділяють цінності відповідальності та довіри. У такій комбінації можна розраховувати на ефективність та взаємну підтримку.

Другий напрямок роботи - навчання громадських активістів та посилення організацій. Багато громадських організацій досі не розуміють, як залучати пересічних громадян до своєї діяльності. Більшість навіть не розуміють, як саме волонтери можуть підсилювати їхню команду. Потрібно допомогти у процесі створення стратегій залучення людей до проектів - формувати мережу активних громадських організацій і професійних координаторів волонтерів.

Третій напрямок - допомога соціальним проектам у містах. Можливість бути посередником між громадськими організаціями та новачками у

громадському секторі. Організація може звертатися по профільну та непрофільну допомогу, а також допомогти знайти людей на проекти.

Четвертий напрямок - громадянська освіта. Це один із важливих інструментів підтримки волонтерської спільноти. Щоб убезпечити волонтерів від вигорання і допомогти з професійним зростанням, важливо не тільки підтримувати різноманітними короткотривалими лайфхаками, а й системно навчати їх. Є багато програм різного рівня - від розвитку особистих навичок до прикладних професійних вмінь.

РОЗДІЛ 1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Огляд існуючих рішень систем реєстрації

Веб-розробка - це робота, пов'язана з розробкою веб-сайту для Інтернету (всесвітньої павутини) або інтрамережі (приватна мережа). Веб-розробка може варіюватися від розробки простої єдиної статичної сторінки простого тексту до складних інтернет-додатків (веб-додатків) для електронного бізнесу та послуг соціальної мережі. Більш повний перелік завдань, до яких часто відноситься веб-розробка, може включати також веб-інжиніринг, веб-дизайн, розро- бку веб-контенту, взаємодію з клієнтами, сценарії на стороні клієнта / сервер, веб-сервер та конфігурація безпеки мережі та розвиток електронної комерції.

Серед веб-професіоналів, веб-розробка, як правило, стосується основних аспектів, не пов’язаних із розробкою веб-сайтів: написання розмітки та кодування. Останнім часом веб-розробка мала на увазі створення систем управління контентом (CMS). Ці CMS можуть бути створені з нуля, фірмового або з відкритим вихідним кодом. Загалом, CMS діє як проміжне програмне забезпечення між базою даних та користувачем через браузер. Основним достоїнством CMS є те, що це дозволяє нетехнічним людям внести зміни до свого веб-сайту без технічних знань.

Для великих організацій та підприємств, команди веб-розробників можуть складатися із сотень людей (веб-розробників) і виконувати стандартні ме- тоди, такі як Agile методології, при розробці веб-сайтів. Менші організації можуть вимагати лише одного постійного або контрагента розробника, або вторинного розпорядження пов’язаними робочими місцями, такими як графічний дизайнер або технік інформаційних систем.

Постійно зростаючий набір інструментів і технологій допомогли розробникам створювати більш динамічні та інтерактивні веб-сайти. Крім того, веб-розробники тепер допомагають надавати програми як веб-сервіси, які традиційно доступні лише як додатки на настільному комп'ютері. Це дало багато можливостей децентралізувати інформацію та розповсюдження ЗМІ. Приклади можна побачити з появою хмарних сервісів, таких як Adobe Creative Cloud, Dropbox та Google Drive. Ці веб-сервери дозволяють користувачам взаємодіяти з додатками з багатьох розташувань, замість того, щоб бути прив'язаними до певної робочої станції для середовища їх застосування.

Приклади драматичних перетворень у сфері комунікації та торгівлі, керовані веб-розробкою, включають електронну комерцію.

Іншим прикладом трансформаційного спілкування під керівництвом веб-розробки є блог. Веб-застосунки, такі як WordPress та Movable Type, створили блог-середовища для сайтів. Зростання використання систем керування контентом з відкритим кодом та CMS підприємства покращило вплив веб-розробки при взаємодії та спілкуванні в Інтернеті.

На сьогоднішній день існує невелика кількість реалізацій онлайн сервісів реєстрації волонтерів. Кожен з них індивідуальний, так як призначений для конкретних цілей. Проаналізуємо аналог веб-сайту у галузі волонтерства: Українська Волонтерська Служба (рис. 1.1).

Інтерфейс сайту максимально простий та зрозумілий, що зменшує кількість помилок при користуванні сайтом. Дизайн виконаний в мінімалістичному стилі, відсутнє нагромадження кольорів та елементів, які б відволікали увагу відвідувачів сайту. Сайт є досить функціональним. На головній сторінці присутні варіанти: стати волонтером або ж знайти волонтерів. Також присутній блог, який інформує про діяльність служби. Присутня можливість підтримки та вже реалізовані прєкти.

Головна сторінка сайту «Українська Волонтерська Служба»:

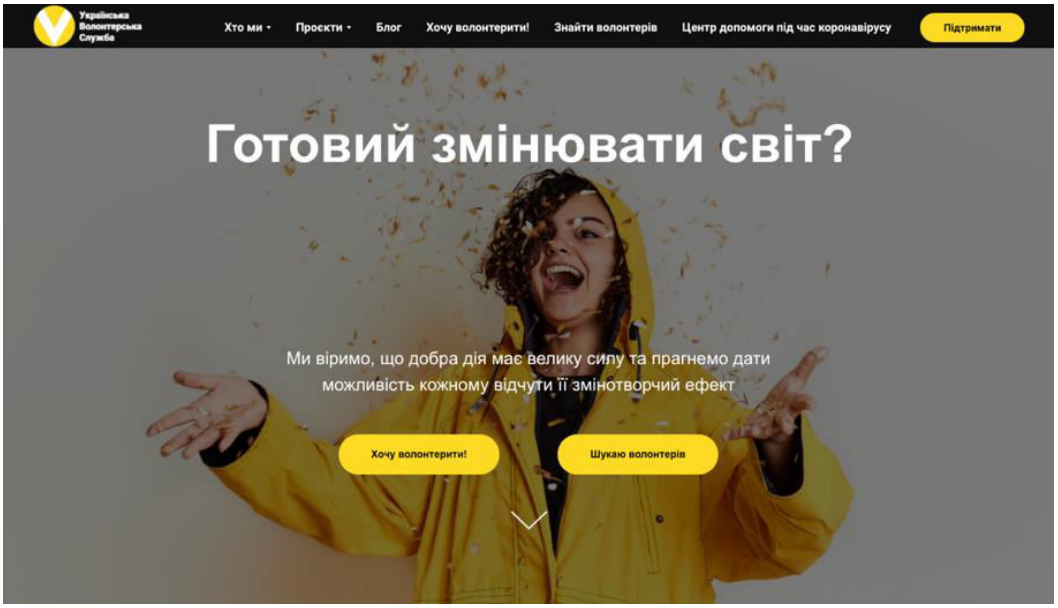

Рисунок 1.1 – Головна сторінка сайту «Українська Волонтерська Служба»

Приклад «звичайної» реєстрації(рис. 1.2). Під «звичайною» мається на увазі реєстрацію на сайті, в обмін на яку отримується логін і пароль, який користувач сам же і ввів. При цьому логіном тут може бути все що завгодно.

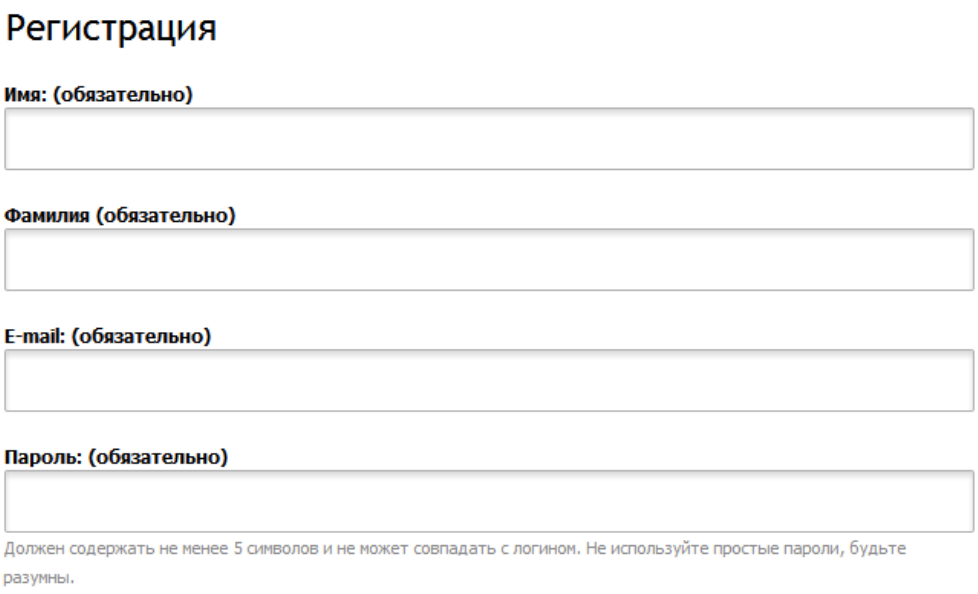

Рисунок 1.2 – Приклад «звичайної» реєстрації

В звичайній реєстрації має бути мінімум полів для заповнення. В ідеалі – логін, пароль і дублікат пароля. Логіном може виступати що завгодно. Найкраще використовувати електронну пошту для середньо і сильно просунутої аудиторії, а також номер телефону для масових сервісів і проектів. Або краще – дати користувачеві вибір. Використовувати в якості логіна ім’я, нік або унікальні ID – не найбільш вдала ідея, яка породжує більше проблем, ніж вирішує.

Реєстрація за допомогою інших ресурсів. Під цим визначенням мається на увазі реєстрацію за допомогою облікового запису в соціальних мережах(рис. 1.3), через ідентифікатор OpenID та окремо, реєстрація з аккаунтом великих систем.

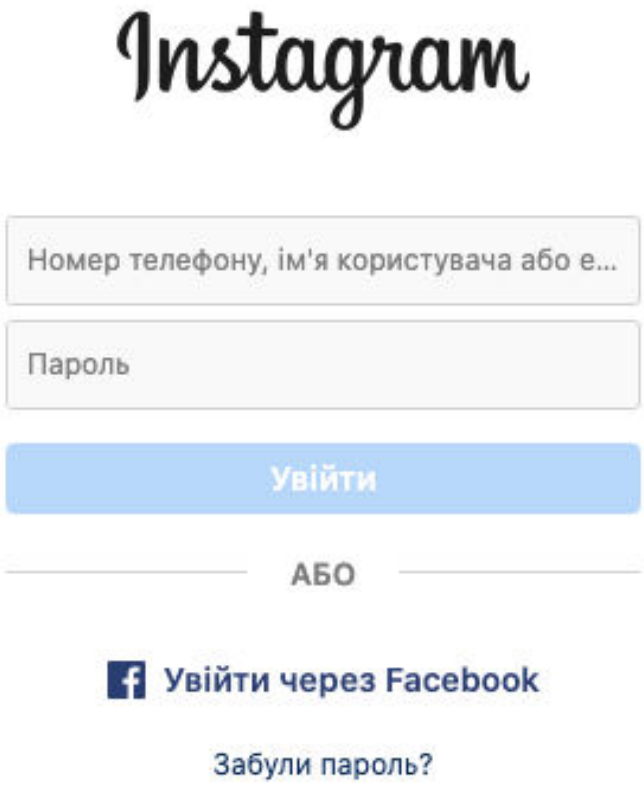

Рисунок 1.3 – Реєстрація за допомогою соціальної мережі

М’ягка реєстрація. Так називається реєстрація, яка не виглядає як окремий процес і відбувається в міру виконання важливих дій. Зазвичай, для м’якої реєстрації потрібен тільки адресу електронної пошти, щоб відправити на неї пароль і запрошення увійти на сайт. Якщо людині це не потрібно, то він просто видалить цей лист або проігнорує його, однак для тих, хто сумнівається це створить ще один шлях входу на сайт, що досить сильно впливає на ефективність реєстрації. Найчастіше її використовують в процесі оформлення замовлення на сайті інтернет-магазинів.

1.2 Переваги вибраної технології для розробки проекту

Ruby on Rails є середовищем, що полегшує розробку, розгортання і обслуговування веб-додатків. За час, що минув з її початкового випуску, Rails пройшла шлях від маловідомої технології до феномену світового масштабу і, що більш важливо, стала саме тим середовищем, яку вибирають, щоб створювати так звані додатки Web 2.0.

Rails добре прижилася з самого початку. Велика кількість розробників була незадоволена тими технологіями, які застосовувалися для створення веб-додатків. І справа, напевно, не в тому, що саме вони використовували – Java, PHP або .NET, - у них накопичувалося відчуття зайвої трудомісткості їх роботи. А потім в один момент прийшла Rails, з якою працювати стало набагато простіше.

Але сама по собі простота не означає спрощеність. Йдеться про професійних розробників, що створюють по-справжньому затребувані у всьому світі веб-сайти. Їм хочеться бачити створені ними програми що витримали випробування часом – спроектованими і розробленими з використанням сучасних, професійних технологій. Тому розробники зайнялися Rails всерйоз і виявили, що вона є не тільки інструментом для розробки вебсайтів. Наприклад, всі Rails-додатки виконуються з

використанням архітектури Модель-Вид-Контролер (Model-View-Controller, MVC). Звична Javaрозробникам середовище виконання, наприклад Tapestry або Struts, теж заснована на MVC. Але Rails йде в використанні MVC ще далі. Rails починається вже з ефектів у програмному забезпеченні, в якому є місце для кожної частини коду, і всі частини вашого застосування стандартним чином взаємодіють один з одним.

Всі Rails-додатки мають вбудоване тестування. У міру додавання до програмного коду, будь-якої функціональної можливості Rails автоматично створює програмні заглушки тестів, призначені для її тестування. Це середовище полегшує тестування своїх додатків, стимулюючи тим самим розробників до цього заняття.

Rails використовує всі можливості Ruby, що полегшує життя програмістів. Програми стають коротшими, читаються легше. Це також дозволяє нам виконувати ті завдання, які інакше виконувалися б в вихідному коді зовнішніх файлів конфігурації. Це полегшує розуміння того, що відбувається.

У Rails-додатках можна побачити лише малу частку повторень,все те що потрібно прописати, прописується тільки в одному місці, яке часто пропонується угодами про MVC-архітектурі, і далі про це можна вже не турбуватися, система дуже продумана в цьому плані.

Для програмістів, які звикли працювати в інших середовищах веб-розробки, де проста зміна може змусити їх вносити в код програми півдюжини, а то і більше правок, це було відкриттям. В Rails практично для кожного аспекту, який зв’язує в єдине ціле ваш додаток, є раціональні умовчання. В Rails можливо написати додаток, використовуючи менше коду, ніж в звичайному веб-додатку, написаному на Java. Якщо потрібно переписати угоди, Rails полегшує і це завдання. Розробники, які переходять на Rails, помічають ще одну особливість. Rails полегшує розробникам інтегрування в їх код таких функцій, як інтерфейси AJAX і RESTful, оскільки

їх підтримка вже вбудована в Rails. Також з Rails можна поширювати вдалу версію свого додатка на будь-яку кількість серверів всього лише однією командою (і так само легко повертати все назад, якщо версія виявиться не зовсім вдалою).

Rails була виділена з реального комерційного застосування. Виявилося, що найкращим способом створення середовища є визначення основних складових конкретного додатка, а потім занесення їх до загального коду. При розробці Rails-додатку в вашому розпорядженні з самого початку вже є половина готового проекту.

1.3 Постановлення задачі

Проаналізувавши можливі варіанти та методи створення вб-сервісу реєстрації волонтерів, можна сформувати наступні задачі для подальшої роботи над проектом.

Перш за все потрібно створити зручну форму реєстрації. Сама реєстраційя буде «нестандартна», без використання бази-даних(рис. 1.4). Метою реєстрації буде збір необхідних даних та надсилання їх менеджеру, який оглядатиме їх, а вже після цього зв’жеться з користувачем для вирішення подальших дій.

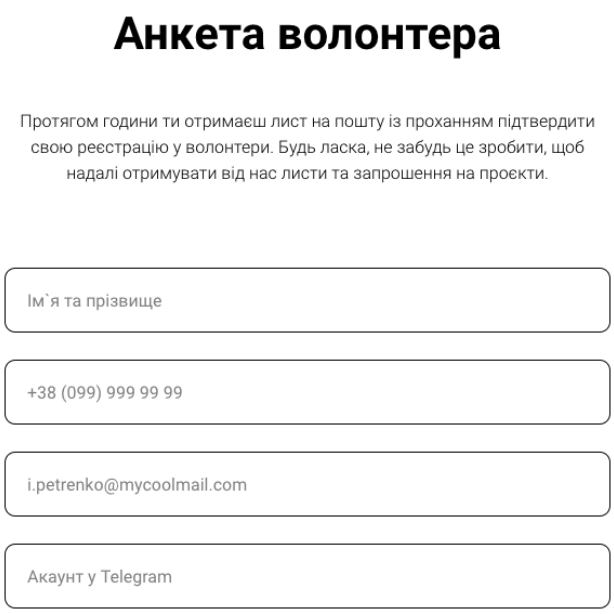

Рисунок 1.4 - Приклад анкети для надсилання даних

Наступним кроком буде створення посилання «About us», де коротко та інформативно описано діяльність, методи роботи даного сервісу та головні цілі.

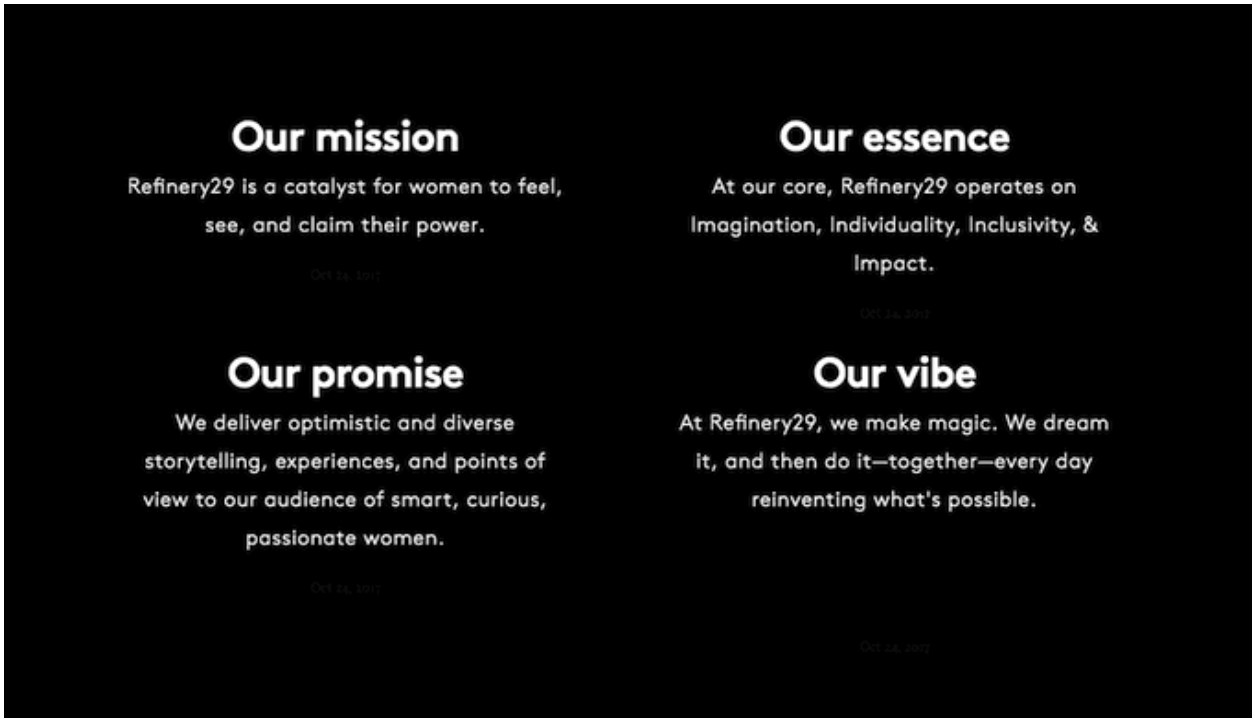

Рисунок 1.5 - Приклад About us сторінки

Також невід’ємною частиною кожного сайту повинно бути header та footer(рис. 1.6). В них повинна міститись навігація по сайту, лого та контакти для фідбеку з користувачами. Інтерфейс повинен бути максимально простим та зручним у використанні.

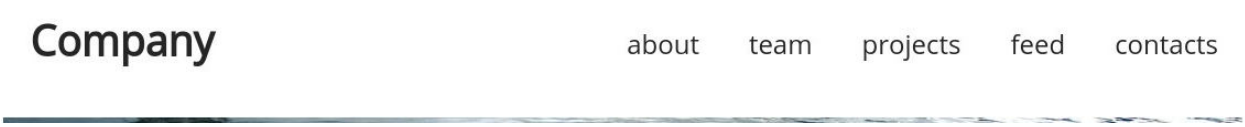

Рисунок 1.6 - Приклад header

Висновки до розділу 1

У першому розділі було розглянуто методи розробки даного проекту, та існуючі аналоги. Описані переваги вибраних технологій, їх актуальність та основи використання. Також описано основні відомості про веб-розробку, її актуальність та найбільш поширені методи розробки.

Проведено аналіз найбільш відомого аналогу, в результаті якого були сформовані вимоги до даного проекту. Сформовано головні задачі які потрібно виконати під час розробки сервісу. Це допомогло реалізувати концепцію створення онлайн сервісу, та запобігти можливим помилкам під час написанню роботи.

РОЗДІЛ 2 ОПИС ПРОЕКТНОГО ТА ТЕХНІЧНОГО РІШЕННЯ

2.1 Основні задачі та методи розробки предметної області

Завдання програми - створити зручний у використанні веб-сервіс реєстрації волонтерів. Для створення даного проекту використовується фреймворк Ruby on Rails.

Ruby - це інтерпретована, повністю об'єктно-орієнтована мова програмування з чіткою динамічною типізацією. Мова вирізняється високою ефективністю розробки програм і увібрала в себе найкращі риси Perl, Java, Python, Smalltalk, Eiffel, Ada і Lisp. Ruby поєднує в собі Perl-подібний синтаксис із об'єктно-орієнтованим підходом мови програмування Smalltalk. Також деякі риси запозичено із мов програмування Python, Lisp, Dylan та CLU.

Багатоплатформова реалізація інтерпретатора мови Ruby поширюється як Вільне програмне забезпечення. Початковий код проекту розповсюджується під ліцензіями BSD ("2-clause BSDL") і "Ruby", яка посилається на останній варіант ліцензії GPL і повністю сумісна з GPLv3.

Для реалізації проекту перш за все необхідно створити головну сторінку та її вигляд. Вигляд повинен бути максимально простий та зручний у використанні для користувача. Далі розробити посилання на форму реєстрації та посилання в якому буде розписано мету та методи роботи сервісу.

В формі реєстрації мають бути присутні:

- прізвище, ім’я;
- номер телефону;
- email;
- дата народження;

RoR створений таким чином, щоб його було дуже легко розширювати за допомогою gem'ів. Це створило велику екосистему ruby gem'ів, яка може розширити ваш додаток і ще більше прискорити процес розробки, скоротивши час, що займається розробкою загальної функціональності. Є декілька найбільш пошерені ruby gems:

Device - це, мабуть, найпоширеніший Gem при використанні Ruby on Rails. Він забезпечує прості у використанні рішення для аутентифікації для програми Rails, який дозволить вам отримати вхід, реєстрацію, забути пароль, блокувати облікові записи та багато інших функцій, пов'язаних з обліковим записом, просто використовуючи цей Gem.

Cells. Часто використовують багато компонентів програми. Зазвичай для цього типу поведінки використовують часткові товари, однак ви повинні переконатися, що контролери, які називаються часткові, мають послідовну поведінку. Cells дозволяє вам взяти частини вашого контролера та інкапсулювати їх у свій маленький контролер. Це допомагає зробити ваш код набагато чистішим та уникнути довгих повідомлень помічників.

Kaminari - це один з найпопулярніших дорогоцінних gem файлів з майже 5 мільйонами завантажень. Він дозволяє вам додати будь-що, від відносин ActiveRecord до простих масивів, використовуючи чистий, простий у використанні та простий API інтерфейс, який є повністю агностичним для будь-якого ORM або шаблону двигуна, який ви використовуєте.

Sidekiq. Існує багато варіантів обробки фону при використанні Ruby on Rails, проте Sidekiq є одним з найпопулярніших. Причиною його популярності є простота API та його масштабність яка набагато краща, ніж інші фонові процесори.

Simple Form. Simple Form має на меті бути максимально гнучким, одночасно допомагаючи потужним компонентам створювати форми (рис. 2.1). Основна мета Simple Form - не змінювати спосіб визначення макету,

дозволяючи вам знайти кращий дизайн. Більшість DSL успадковані від Formtastic.

Simple Form була розроблена так, щоб її налаштування були такими, як вам потрібно. В основному це стек компонентів, які викликаються для створення повного вводу html, який за замовчуванням містить мітку, підказки, помилки та сам вхід. Він не має на меті створювати багато різних логік які вже присутні в Rails. Натомість Simple Form діє як DSL і просто відображає ваш тип введення (отриманий з визначення стовпця в базі даних) на конкретний допоміжний метод.

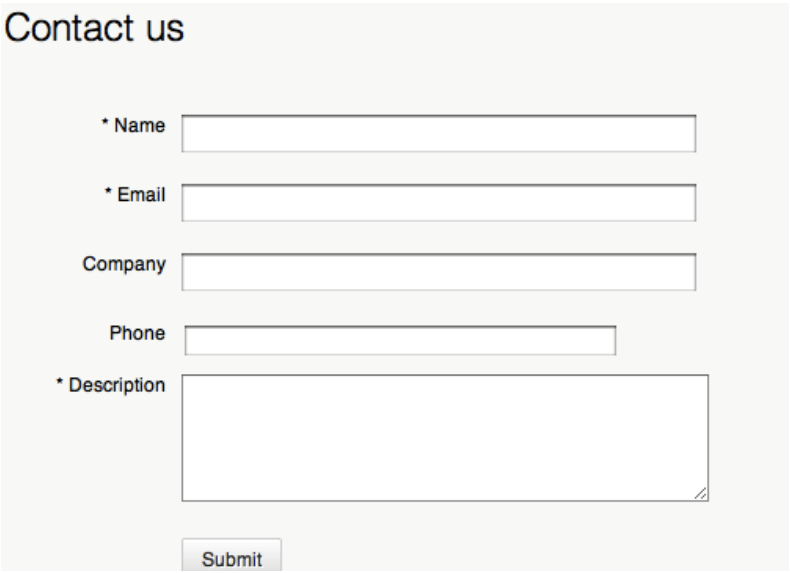

Рисунок 2.1 - Приклад «Contact us» форми Simple Form

Devise - це ruby-gem, що надає можливості для аутентифікації в rails-додатках (рис. 2.2). Devise працює в зв'язці з гемом Warden, який в свою чергу надає сам механізм для аутентифікації в rack-базованих ruby-додатках (рис. 2.3). Основні особливості Devise описані нижче:

заснований на Rack;

є закінченим MVC-рішенням, заснованим на Rails;

дозволяє вхід в систему по декількох моделях одночасно;

заснований на модульній: використовує тільки те, що вам дійсно необхідно.

```
<p class="navbar-text float-right">
<% if user_signed_in? %>
  Logged in as <strong><%= current_user.email %></strong>.
  <%= link_to 'Edit profile', edit_user_registration_path, :class => 'navbar-link' %> |
  <%= link_to "Logout", destroy_user_session_path, method: :delete, :class => 'navbar-link'  %>
<% else %>
  <%= link_to "Sign up", new_user_registration_path, :class => 'navbar-link'  %> |
  <%= link_to "Login", new_user_session_path, :class => 'navbar-link'  %>
<% end %>
</p>
```

Рисунок 2.2 – створення форми реєстрації gem Device

Рисунок 2.3 - вигляд форми реєстрації gem Device

Деякі особливості Ruby:

- вільне форматування - ви можете починати писати код з будь-якого рядка чи колонки;

- коментарі- все, що не є в лапках і стоїть після "#", ігнорується інтерпретатором до кінця рядка. Якщо ви бажаєте залишити коментар в декілька рядків, то почніть перший рядок з "=begin", а останній - з "=end";

- розділення операторів - декілька операторів, які записані в один рядок, повинні бути розділені крапкою з комою (;), але вона не потрібна

наприкінці рядка, який сам інтерпретується, як крапка з комою. Якщо така поведінка є небажаною, то треба ставити наприкінці рядку зворотний слеш;

-Ключові слова - також відомі як зарезервовані слова (близько 41-го), як правило, не можуть бути використані для цілей, які відрізняються від вбудованих.

Можливо ви звикли до того, що false може бути представленим у вигляді нуля, порожнього рядка чи нульового символу, але в Ruby всі ці значення інтерпретуються як true; в Ruby все true, крім зарезервованих слів і нуля.

У більшості мов ключові слова можна було б назвати "зарезервованими словами" через заборону на використання їх в якості назв для ідентифікаторів. Парсер Ruby є гнучким і не буде скаржитися, якщо ви будете використовувати ключові слова для імен класів чи глобальних змінних, додаючи до них префікси "@", "@@" або "$".

В Ruby числа без дробової частини інтерпретуються як integer, а з нею як float. Integer - буквально послідовність цифр, такі, як 0, 123 або 0987654321. Нижнє підкреслення ігнорується, якщо воно є у складі integer-числа, але не стоїть на початку або наприкінці. Це іноді використовується для розподілу числа, наприклад, як роздільник тисяч: 1_000_000_000. Ось листинг програми p002rubynumbers.rb (рис. 2.4):

```
# p002rubynumbers.rb
=begin
 Числа Ruby
 Стандартні оператори:
 + додавання
 - віднімання
 * множення
 / ділення
=end

puts 1 + 2
puts 2 * 3
# Цілочисельне ділення
# Коли ви виконуєте арифметичні операції з цілими числами, ви отримуєте ціле число в якост
puts 3 / 2
puts 10 - 11
puts 1.5 / 2.6
```

Рисунок 2.4 – Оператори Ruby

Integer є об'єктом класу Fixnum або Bignum в залежності від того, якого він розміру. Floating-point числа належать до класу Float, який відповідає реалізованому архітектурно double-data типу.

Класи Complex, BigDecimal та Rational не є вбудованими в Ruby, але постачаються разом зі стандартною бібліотекою. Ми поговоримо докладніше про класи у наступних статтях.

Оператори та порядок виконання (рис 2.5). У цій таблиці вони відсортовані за своїм пріоритетом:

Метод	Оператор	Опис
•	[] []=	Посилання на елемент, набір елементів
•	**	Піднесення до степеня
•	! ~ + -	Заперечення, доповнення, унарні плюс та мінус (методи для

Рисунок 2.1 – Відсортовані за приорітетом оператори Ruby

Оператор ділення з остачею в Ruby (рис. 2.5).

```
puts (5 % 3)     # результат: 2
puts (-5 % 3)    # результат: 1
puts (5 % -3)    # результат: -1
puts (-5 % -3)   # результат: -2
```

Рисунок 2.6 – Приклад оператора ділення з остачею

Визначення цього оператору в Ruby відрізняється від аналогічних в C та Java: 7%3=2, коли в C або Java ми б отримали б "-1" в якості відповіді. В Ruby знак результату для оператора ділення з остачею завжди такий самий як знак другого операнда.

Основною перевагою мови програмування Ruby і фреймворка Ruby on Rails вважається швидкість розробки. Практика показує, що швидкість

розробки проектів на RoR збільшується на 30-40 відсотків по відношенню до будь-якої іншої мови програмування або фреймворку. В першу чергу приріст швидкості розробки визначається великим набором готових до роботи штатних інструментів RoR, колосальним набором готових рішень в співтоваристві, мови Ruby і простоті програмування на ньому.

Однією з найважливіших частин культури RoR є соціальність. Вирішив проблему, - допоможи вирішити іншим. Реалізував свій модуль, - поділися з спільнотою. Таким чином, на даний момент у відкритому доступі зберігаються тисячі готових рішень тих чи інших завдань. Системи аутентифікації, авторизації, коментування, системи платежів, поштові розсилки та багато іншого (все те, що зазвичай розробляється «з нуля») впроваджуються реалізовані кимось іншим, протестовані і рекомендовані численним співтовариством.

2.2 Розробка Use Case діаграми.

За допомогою Use Case можна виявити функціональні вимоги до сервісу: практично кожен рядок Use Case є окремою функціональною вимогою. Можна побачити, які функції повинні виконуватися разом, а отже, з'являється можливість виставляти пріоритети реалізації цих вимог так, щоб вони були готові в один час.

Звичайно, для розробки функціональних вимог до системи потрібно написати цілий набір Use Case, який враховує цілі користувачів кількох ролей. Цей набір дозволяє забезпечити повноту вимог користувачів до системи. Набір Use Case є набором вимог вищого рівня абстракції, ніж набір окремих функціональних вимог, і в той же час повністю покриває призначені для користувача вимоги до функціональності. Тому він більш зручний в роботі.

Розглянемо Use Case написаний для веб-сервісу реєстрації волонтерів для інформаційно-розважальних подій (рис. 2.7):

Дійові особи - користувач, система.

Цілі - користувачу заповнити та відправити форму, системі відправити форму менеджеру.

Вдалий сценарій:

користувач заходить на сторінку реєстрації;

користувач вводить необхідні дані та відправляє їх;

система перевіряє на валідність форму;

система відправляє форму менеджеру;

система видає користувачу повідомлення про успішну реєстрацію.

Результат - користувач успішно відправив дані для подальшої участі в волонтерстві.

Рисунок 2.7 – Use Case діаграма веб-сервісу

Ruby on Rails - це фреймворк. Найчастіше фреймворк не дозволяє вам самодіяльність. Звичайно ж, в Ruby on Rails можна «винайти свій велосипед»

і програмувати в будь-яких напрямках, не спираючись на стандарти; але часто цього не потрібно. Стандарти розміщення файлів в проекті, стандарти написання коду в проекті, загальні правила програмування в Ruby on Rails сильно структурують будь-який проект. За рахунок цього проект стає читаним.

Входження в проект новачків відбувається дуже швидко. Досвід показує, що будь-який новачок в проекті в перший же день роботи робить свої перші корисні правки. За рахунок цього не вважається великою проблемою, якщо розробку проекту спочатку вела одна команда програмістів, а підтримку проекту або доопрацювання - зовсім інша. Проект на RoR апріорі зрозумілий будь-якому розробнику.

При розробці будь-якого великого проекту постає резонне питання. Як і хто буде тестувати проект? Не завжди є кошти і бажання створювати цілі відділи тестування, до того ж хочеться автоматизувати цей процес. На відміну від інших фреймворків, в складі RoR є великі гроші автоматизованого тестування. В інших мовах програмування і фреймворки штатних засобів тестування немає. Звичайно, є сторонні розробки, що дозволяють організувати автоматичне тестування проекту на PHP, але вони не ставляться "з коробки" і про їх використання програмісти частіше не замислюються.

У проекті на Ruby on Rails, в ідеалі, код проекту не пишеться до тих пір, поки під цей код не написані тести. RoR ідеологія передбачає початкове використання методів BDD (Behavior Driven Development) або TDD (Test Driven Development).

Кешування проектів - один з найважливіших етапів розробки великого інтернет-проекту. У PHP є різні варіанти кешування даних. Ці варіанти і інструменти прикручуються, прилаштовуються, прилаштовується, прикріплюються збоку. До сих пір в співтоваристві PHP немає єдиної думки:

що краще використовувати, як краще кешувати дані, якими інструментами користуватися.

Ruby on Rails в його базової комплектації має штатні засоби кешування даних. На старті надаються інструменти, що дозволяють реалізувати кешування даних на проекті. Ви можете кешувати цілі сторінки або ж блоки коду. Можете кешувати результати запитів і ActiveRecord-моделі. Кешувати можна як за допомогою memcached або redis, так і іншими засобами. Для реалізації кешування на Ruby on Rails проект вам в 95 відсотках випадків не буде потрібно нічого крім уже готових і штатних рішень.

Часто зустрічається ситуація, коли хтось зробив проект, а потім несподівано розуміє, що для продовження розвитку проекту необхідна англійська версія. Розробники на PHP при цьому починають заводити розмови про те, що це не було передбачено заздалегідь, що це довго і вкрай складно. Давайте, мовляв, відкриємо паралельний проект, який буде повною копією цього, і переведемо його.

Ruby on Rails в базовій комплектації має засоби локалізації проекту. Ви можете передбачити необхідність підтримки різних мов на сайті як спочатку, так і в подальшому. RoR вміє роздавати різні шаблони для різних мов, містить в собі конфігураційні файли з перекладами термінів і багато інших штатні інструменти для реалізації локалізації проекту.

Найчастіше в багатьох PHP проектах ми можемо бачити картину, коли адреса певної сторінки величезна і незрозуміла. В Ruby on Rails є штатна можливість гнучко налаштувати ваш роутинг, вид адрес, назви основних розділів. Є можливість швидко змінити адреси в одному місці без необхідності зміни цієї адреси в усьому проекті. У співтоваристві RoR-розробників активно використовуються ідеологія REST. Адреси сторінок в проектах на Ruby on Rails завжди зрозумілі, красиві, чудово розуміються пошуковими системами, прості.

Валідації в ruby on rails прекрасно реалізовані інструменти, що дозволяють затверджувати вхідні дані. Ваші користувачі заповнюють форми і потрібно перевірити правильність введення адреси електронної пошти, наявність пароля або необхідну мінімальну довжину логіна, - штатні засоби Rails вам в цьому допоможуть.

На даний момент продуктивність Ruby не поступається PHP. Але чи так це важливо? Адже час генерації сторінки, в основному - це час, витрачений на запити в базу даних. Швидкість самого мови зазвичай не грає велику роль.

При цьому у вас є можливість користуватися головною перевагою RoR - швидкістю розробки проектів і низькою вартістю їх підтримки. В даний момент вартість розробників на порядок дорожче вартості зайвої планки пам'яті в сервер. У будь-якому випадку, проблеми продуктивності будь-якого проекту, - це не проблеми невірного вибору платформи або мови програмування. Швидше за все, це проблеми помилковою архітектури проекту, кешування даних або оптимізації БД.

2.3 Діаграма класів сервісу реєстрації

Діаграма класів - статичне представлення структури моделі. Відображає статичні елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти, а також їхні відносини.

Асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності.

Якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарною. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів; вони називаються n-арними. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносин. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це «обличчя», яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Ви можете явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді. Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: «нуль або одиниця» (0..1), «багато» (0 .. *), «одиниця або більше» (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За

допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає «будь-яке число об'єктів, крім 2 і 5».

Агрегація - проста асоціація між двома класами, яка відображає структурне відношення між рівноправними сутностями, коли обидва класи знаходяться на одному концептуальному рівні, і жоден з них не важливіший за решту. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має вищий ранг (ціле) і складається з декількох менших за рангом (частин). Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і її зображуєть як просту асоціацію з незафарбованим ромбом з боку «цілого».

Графічно агрегація представлена порожнім ромбом на блоці класу, і лінією, яка проведена від цього ромба до класу, що міститься в ньому.

Композиція - більш строгий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та екземплятів класів що містяться в ньому. Якщо контейнер буде знищений, то весь його вміст буде також знищено.

Технічно файл з розширенням .gem є звичайним архів, у якому перебуває файл специфікації і вихідний код бібліотеки в стані на момент релізу.

У специфікації міститься досить багато інформації, але найголовніше:

- назва і версія даного gem'a;
- назви та версії gem'ов, без яких робота буде неможлива;
- дані про автора і опис gem'a.

Крім бібліотечних файлів, які ви підключаєте в коді свого застосування, до складу gem'a можуть входити виконувані файли, які після установки «видно» на системному рівні. Насправді це не бінарні, а текстові файли - програми, написані на Ruby. Коли ви запускаєте їх, ОС викликає

інтерпретатор ruby, який і займається їх виконанням. Повинно бути, найвідоміші виконувані файли - це rails, rake і gem.

Найбільш популярний фреймворк для Ruby - Rails. Він забезпечує приблизно таку ж функціональність для мови, як Spring для Java.

Одна з речей, яка всім подобається в Rails, - потрібен лише один файл для оголошення всіх своїх endpoints. Завжди можна використовувати команду терміналу $ rake routes для того, щоб їх побачити. Це відмінний варіант для великих проектів, коли вам необхідно зробити щось на основі того, що вже написано.

Крім того, ви можете розділити ваші endpoints на групи. Наприклад, коли у вас є модель User, ви можете встановити шляхи для всіх її членів таким чином, щоб кожен endpoint автоматично отримував свій ідентифікатор.

Вам не доведеться використовувати різні параметри для тих же endpoints. У Rails, за замовчуванням, ви можете передавати будь-які параметри в кінцеву точку і просто проводити валідацію для тих з них, які ви хотіли б використовувати в контролері.

Ruby on Rails - об'єктно-орієнтований програмний каркас (фреймворк) для створення веб-застосунків, написаний на мові програмування Ruby. Ruby on Rails надає каркас модель-вид-контролер (Model-View-Controller) для веб-застосунків, а також забезпечує їхню інтеграцію з веб-сервером і сервером бази даних.

Ruby on Rails був створений Девідом Гайнемаєр Генссоном на основі його роботи над засобом керування проектами Basecamp і був випущений в липні 2004 року. Ruby on Rails є відкритим програмним забезпеченням і розповсюджується за ліцензією MIT.

Основними компонентами застосунків Ruby on Rails є модель (model), вид (view) і контролер (controller).

Модель надає решті компонентів програми об'єктно-орієнтоване представлення даних (таких як каталог продуктів або список замовлень).

Об'єкти моделі здійснюють завантаження і збереження даних в реляційній базі даних.

Завдяки можливостям динамічної типізації в мові Ruby розробникові досить успадкувати свій клас моделі від базового класу ActiveRecord::Base. Ruby on Rails автоматично пов'язує класи моделі з таблицями в базі даних і створює атрибути об'єктів для відповідних полів таблиці.

Вид створює інтерфейс користувача для відображення отриманих від контролера даних. Вид також передає запити користувача на маніпуляцію даними в контролер (як правило, вид не змінює безпосередньо дані з моделі).

У Ruby on Rails вид описується за допомогою шаблонів RHTML. Вони є файлами HTML з додатковими включеннями фрагментів коду Ruby (Embedded Ruby або ERb). Вивід, згенерований вбудованим кодом Ruby, включається в текст шаблону сторінки HTML, яка після цього повертається користувачеві. Види можуть використовувати фрагменти інших видів і, у свою чергу, бути включеними в шаблон (layout) вищого рівня.

Контролер - основний компонент, що відповідає за взаємодію з користувачем. Контролер прочитує необхідні дані з моделі і готує їх для відображення, а також зберігає отримані від відображення дані в моделі.

Контролером в Ruby on Rails є клас, успадкований від ActionController::Base. Відкриті методи контролера є так званими діями (actions). Action часто відповідає окремому видові. Наприклад, по запиту користувача admin/list буде викликаний метод list класу AdminController і потім використаний вид list.rhtml.

Всі міграції тут прописані в додатку, тому настройка бази даних на різних пристроях зводиться до виконання однієї команди: $ bundle rake db: setup. Таким чином, зовнішній клієнт для настройки або використання бази даних просто не потрібен.

І ні, база даних, яку ви створили і перенесли на інший пристрій, не буде порожній: в вашому Rails-додатку є файл з ім'ям seeds.rb, в якому ви можете

вказати всі записи для різних моделей, необхідних для роботи програми. У підсумку на модель потрібно всього лише кілька рядків коду.

Команда $ bundle rake db: setup виконує три функції:

- створює базу даних, якщо її ще немає;
- запускає всі міграції;
- заносить всі вихідні дані з вашого seed-файлу.

Код дійсно чистий: в ActiveRecords просто записуються методи, які вам необхідно реалізувати, а не атрибути.

У фреймворку Hibernate для Java необхідно прописувати всі атрибути з анотаціями, а потім ще й сеттери для тих атрибутів, які вам необхідно модифікувати. В цьому випадку ви отримуєте на виході велику кількість коду.

У Rails те ж саме займає один рядок. DB Schema зберігається в файлі schema.rb, який автоматично створюється при запуску міграції. І в класі не потрібні сеттери або атрибути. Коли будуть потрібні останні, досить буде написати: Model.attribute - і це все.

Як вже говорилося вище, з Rails ви можете фокусуватися саме на логіці і методах вашого проекту, а не на коді.

Навколо Ruby on Rails склалася велика екосистема додаткових плагінів з відкритим вихідним кодом («джемів», gems), які реалізують найбільш затребувані функції (рис. 2.8).

«Джеми» бувають дуже різні: від низькорівневих, що відповідають за якийсь аспект внутрішньої роботи програми, до високорівневих, що представляють собою окремі модулі для вирішення цілого спектру бізнес-завдань. Використання системи подключамих плагінів багато в чому і послужило причиною високої популярності фреймворка - можливість вибірково підключати окремі компоненти і бібліотеки дуже сильно прискорює розробку, а той факт, що використовувані розширення добре протестовані роками, забезпечує надійність рішень.

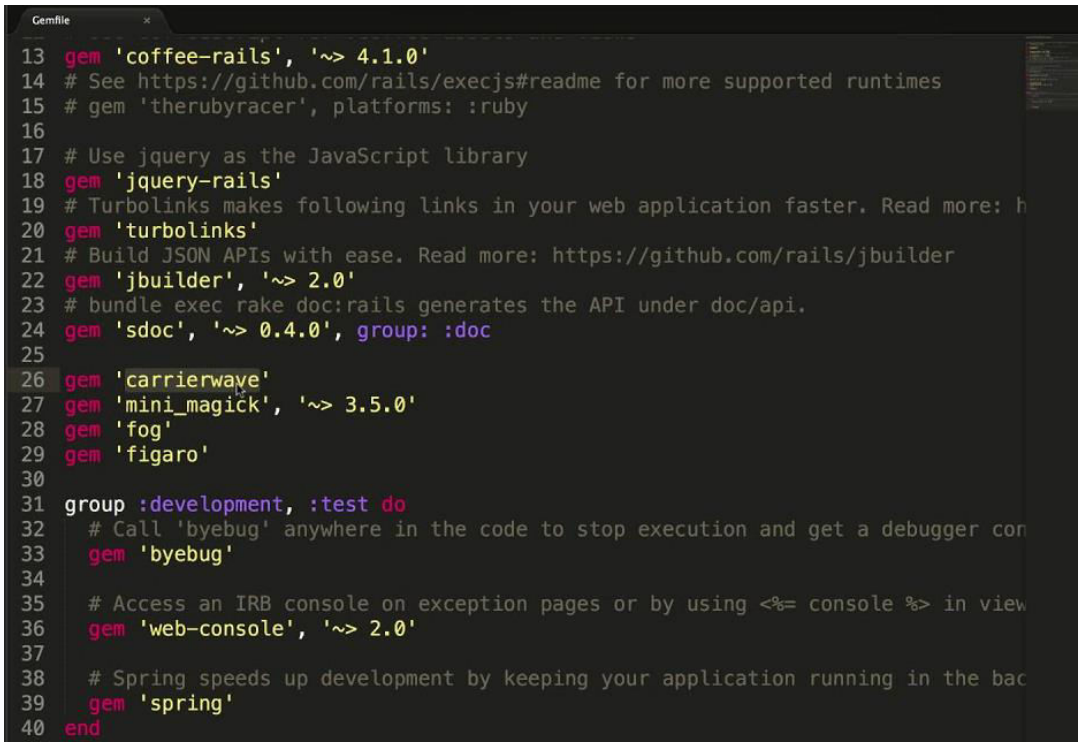

Рисунок 2.8 – Gem файли Ruby on Rails

Ruby on Rails (RoR) - фреймворк, написаний на мові програмування Ruby, що дозволяє розробляти надійні і супроводжувані веб-додатки під високі вимоги до швидкості роботи і стійкості до навантажень.

Фреймворк Ruby on Rails використовувався при створенні таких популярних сайтів і додатків, як inSales, Shopify, Lenta.ru, Netflix, Basecamp, GitHub, Zendesk, Twitter, SoundCloud, Airbnb, Diaspora, Groupon, Dribbble, Bloomberg, Hulu, Kickstarter, Yellow Pages , Change.org, SlideShare, Upwork, 500px, Couchsurfing і багатьох інших. Всі ці проекти досить відомі і відвідувані.

Ruby on Rails написаний на мові програмування Ruby, тобто до фреймворку також застосуємо набір принципів Ruby Way:

Програмування, орієнтоване на людину, а не на комп'ютер. Ruby створений для написання програм в першу чергу зрозумілих людині, а лише потім - комп'ютера. Будь-яка робота з комп'ютером виконується людьми і для

людей, тому необхідно піклуватися в першу чергу про витрачених зусиль людей, а не про економію декількох байт пам'яті або тактів процесора. У сучасному світі обчислювальні ресурси значно дешевші часу розробників, тому з точки зору бізнесу цей підхід теж виправданий.

«Принцип найменшого подиву» - в ергономіці цей принцип означає, що якщо призначення якогось елементу неясно, то його поведінка повинна бути найбільш очікуваним з боку користувача. У перекладенні на програмування цей принцип звучить так: «Програма повинна вести себе так, як очікує програміст». Ruby і Ruby on Rails дотримуються цього патерни проектування, що істотно заощаджує час і нерви розробників.

«Просто, але не дуже просто» - стислість вітається, але не на шкоду зрозумілості; тобто і надмірність допустима, якщо вона виявляється зручною (наприклад, логіка програми може бути більш зрозумілою завдяки більшій багатослівності).

Прості строгі правила, виконання яких не доходить до педантизму. Правила і угоди дуже корисні в розробці, так як написаний за правилами код простіше і швидше розуміється іншими людьми. При цьому, в деяких контекстах проходження суворим універсальними правилами цілком може бути абсурдним. Мова Ruby і фреймворк Ruby on Rails містять в собі багато правил і угод, але при цьому дають розробникам можливість їх порушити, якщо розробник вважатиме це виправданим.

Зараз в веб-розробці використовується досить багато фреймворків, але ми в якості основного вибрали саме Ruby on Rails.

Цей вибір цілком усвідомлений і він заснований на досвіді роботи з декількома платформами. Зокрема, ми починали з фреймворків Yii і Symfony, потім досить багато працювали з Laravel і Django - все це хороші фреймворки, але в наших кейсах Ruby on Rails виявився більш оптимальним рішенням як по швидкості і якості розробки, так і за рівнем подальшої

сопровождаемости.Багато популярних UNIX-подібні ОС поставляються з прийнятною версією SQLite3.

SQLite - полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92. Сирцевий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою.

Rails поставляється з рядом скриптів, названих генераторами, розроблених для полегшення життя розробника, створюючи все, що необхідно для початку роботи над певним завданням. Одним з них є генератор нової програми, що надає вам основу програми Rails, таким чином, вам не потрібно писати його самим (рис. 2.9).

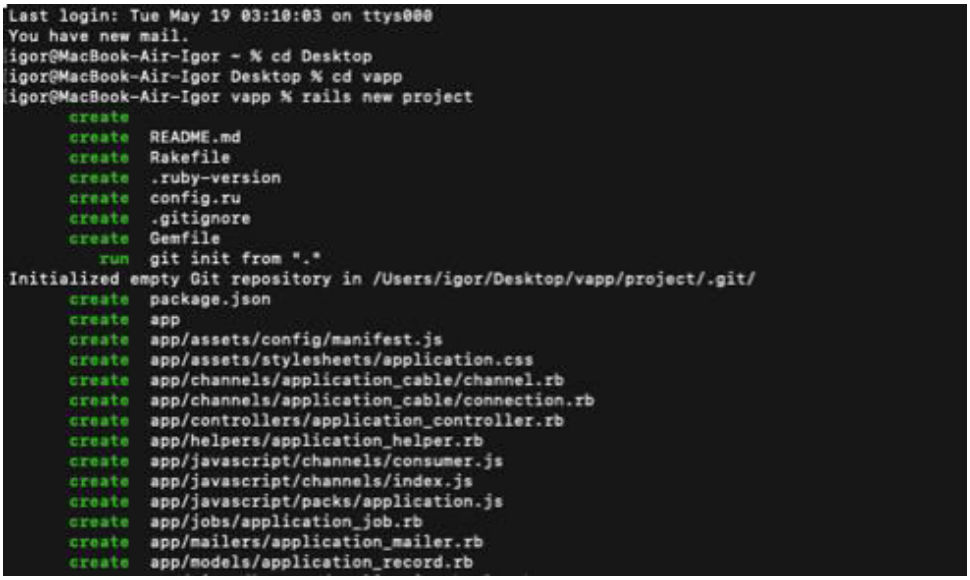

Рисунок 2.9 - створення нового проекту Rails

В директорії project є кілька автоматично згенерованих файлів і папок, які задають структуру програми на Rails.

Bundler. Що таке Bundler? Це менеджер для управління залежностями gem в ruby додатках. Ця утиліта дозволяє легко встановлювати необхідні gem'и для вашого застосування, при цьому зовсім не залежати від встановлених в системі. Якщо ви використовували Rails для своїх розробок,

то згадайте, як задавали залежності gem'ів за допомогою config.gem в enviroment.rb, Bundler вирішує цю задачу набагато зручніше і простіше. Його включили в Rails 3.0 за замовчуванням і тепер, саме він використовується для управління залежностями gem'ів в даній версії фреймворка. Цю утиліту можна використовувати для будь-якого ruby фреймворка.

Ruby on Rails - надзвичайно потужний інструмент для розробки веб-додатків. Він оснащений великою кількістю вбудованих функцій, які допомагають прискорити розвиток веб-додатків, таких як інтелектуальна маршрутизація та відображення відношення об'єктів, використовуючи шаблон MVC.

Model-View-Controller (MVC, «Модель-Представлення-Контролер», «Модель-Вид-Контролер») - схема поділу даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: модель, уявлення і контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно .

Модель (Model) надає дані і реагує на команди контролера, змінюючи свій стан.

Подання (View) відповідає за відображення даних моделі користувачеві, реагуючи на зміни моделі .

Контролер (Controller) інтерпретує дії користувача, сповіщаючи модель про необхідність змін .

Концепція MVC (рис. 8) була описана Трюгве Реенскаугом в 1978 році, який пра- цював в науково-дослідному центрі «Xerox PARC» над мовою програмування «Smalltalk». Пізніше, Стів Бурбек реалізував шаблон в Smalltalk-80.

Остаточна версія концепції MVC була опублікована лише в 1988 році в журналі Technology Object. Згодом шаблон проектування став еволюціонувати. Наприклад, була представлена ієрархічна версія HMVC; MVA, MVVM. Подальший виток популярності привнесло розвиток

фреймворків, оріє- нтованих на швидку розгортку, на мовах Python і Ruby, Django і Rails. На мо- мент 2017 року, фреймворки з MVC зайняли помітні позиції по відношенню до решти фреймворками без цього шаблону.

Шаблон проектування MVC передбачає поділ даних програми, призначеного для користувача інтерфейсу і керуючої логіки на три окремих компоненти: Модель, Представлення і Контролер - таким чином, що модифікація кожного компонента може здійснюватися незалежно.

Термін «компонент» в даному випадку не має ніякого зв'язку з компонентами деяких популярних CMS або фреймворків, а компоненти Бітрікс, взагалі будуються з усіх трьох складових MVC.

У наведеному визначенні під компонентом слід розуміти якусь окрему частину коду, кожна з яких грає одну з ролей Контролера, Моделі або Уявлення, де Модель дозволяє отримувати і керувати даними додатка, Представлення відповідає за видиме користувачеві відображення цих даних (тобто, в застосуванні до інтернету формує відсилаючий сервером браузеру користувача HTML / CSS), а Контролер керує всім цим оркестром.

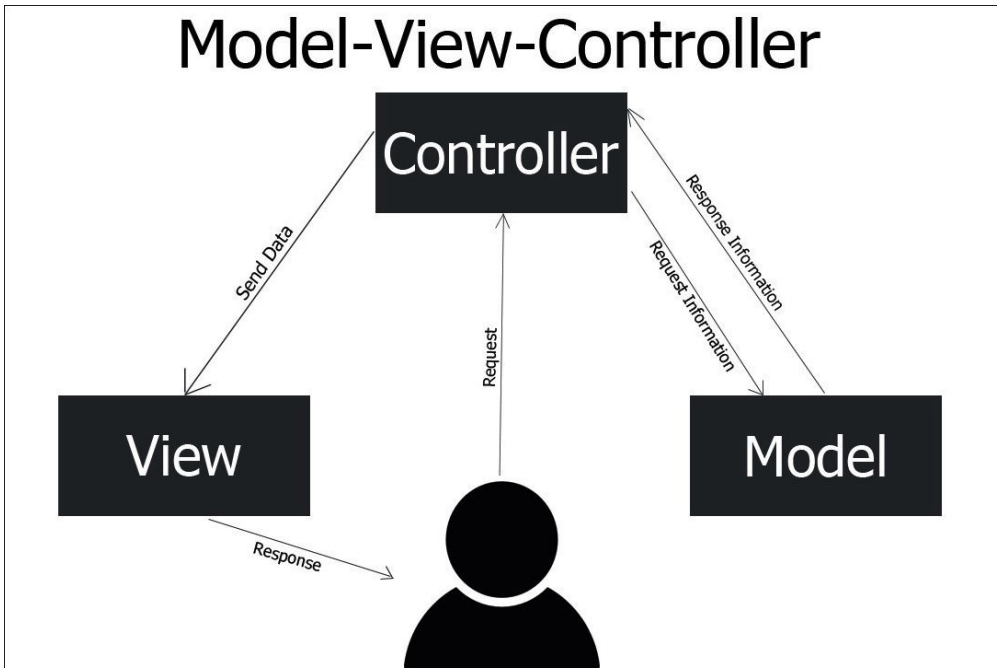

Рисунок 2.10 – Концепція MVC

На рисунку 8, пунктирними лініями показана керуюча інформація (така, наприклад, як ID запитуваних записів блогу або товару в магазині), а суцільними - власне дані додатку (які можуть зберігатися в БД, або у вигляді файлів на диску, або навіть, можливо , в оперативній пам'яті - це питання лежить за межами паттерна MVC). У застосуванні до Інтернету запит і відповідь ходять по HTTP, тому можна умовно вважати, що на цьому малюнку позначені заголовки HTTP-запиту і відповіді.

Отримавши запит, контролер його аналізує, і в залежності від результатів обробки може видати такі варіанти відповідей:

1. Відразу видати відповідь про помилку (наприклад, при запиті неіснуючої сторінки віддати тільки HTTP-заголовок «404 Not found»)
2. Якщо надійшов запит визнаний коректним, то, в залежності від того, є він запитом на перегляд або на модифікацію даних, контролер викликає відповідний метод моделі, такий як Save або Load

Важливе зауваження: концепція MVC не тільки не прив'язана до якоїсь конкретної мови програмування, вона також не прив'язана і до використовуваної парадигмі програмування. Тобто, ви цілком можете проектувати свій додаток по MVC, при цьому не застосовуючи ООП, і спроектувати Модель Товар для інтернет-магазину таким чином:

```
<?
mixed Product_Load (int $id) { ... }
// повертає асоціативний масив з даними про товар або FALSE при невдачі

bool Product_Save (array $data) { ... }
// повертає TRUE при вдалому збереженні даних $ data, або FALSE при
невдачі
?>
```

в залежності від отриманого від моделі відповіді контролер вирішує, який з представлень викликати для формування підсумкової відповіді на початковий запит.

Питання про те, хто повинен перевіряти на валідність і права доступу вхідні дані (Контролер або Модель), є предметом досить численних суперечок, оскільки патерн MVC не описує таких деталей. Це означає, що в цьому питанні вибір за вами (або за вас його зробили автори вашого улюбленого фреймворка або CMS).

Контролер перевіряє вхідні дані на предмет «загальної» (тобто незалежної від конкретного запиту) коректності, відповідність вимогам Моделі до валідності даних, що зберігається перевіряє відповідний метод Моделі, а права доступу метод Access окремого класу User.

Для виклику Уявлення в PHP іноді проектується спеціальний клас (а то і кілька класів), наприклад, View (часто зустрічається в описах MVC в реалізації того чи іншого фреймворка), проте це не є вимогою MVC.

Також файли уявлень часто називають шаблонами, а при використанні так званих шаблонизаторів (рис. 2.11) роль Уявлення грає сам шаблонизатор, а шаблони (тобто файли, що містять безпосередньо HTML-розмітку) в деяких фреймворками називають layouts.

```
<!-- HTML.header -->

<h1><?=$product->Title;?></h1>
<p>Цена:<b class="price"><?=$product->Price;?></b></p>
<p class="description"><?=$product->Description;?></p>

<!-- HTML.footer -->
```

Рисунок 2.11 – View файл шаблону product.tpl.php

Шаблон product.tpl.php відображає дані про товар (які до моменту його виклику вже містить об'єкт $ product).

```
<!-- HTML.header -->

<h1 class="error">Ошибка: <?=$error;?></h1>

<!-- HTML.footer -->
```

Рисунок 2.12 – Повідомлення про помилку

Шаблон error.tpl.php (рис. 2.12) відображає повідомлення про помилку (яке міститься в змінній $ error)32

Контролер product.php, службовець для отобоаженія Товару, буде виглядати приблизно так:

```
<?

include 'product.class.php';
// в цьому файлі декларуються методи Моделі

// визначення цієї функції в контролері, звичайно, неправильно
// в даному випадку вона тут тільки для наглядності

function Error ($error) {
            // виводить повідомлення про помилку і завершує роботу
            контролера, приблизно так:

             header('Правильний статус помилки, наприклад, 400 або 404');
             $error = 'Відповідне повідомлення користувачу, наприклад,
             Сторінки не існує';
              include 'error.tpl.php'; // шаблон для відображення помилки
              exit;
}
```

```
if (!$id = ...) // перевірка "загальної" валідності Запиту 1
      error(...);


// перевірка прав доступу

if (!$user->Access(...))
      error(403);


if (!$product = Product::Load($id)) // Запрос 2 та аналіз Відповіді 2
      error('Тут скорее всего случилась ошибка БД');

include 'product.tpl.php'; // Запрос 3 та Відповіді 3 та 4

?>
```

Відмінності опису концепції шаблону.

З розвитком об'єктно-орієнтованого програмування і поняття про шаблони проектування - був створений ряд модифікацій концепції MVC, які при реалізації у різних авторів можуть відрізнятися від оригінальної. Так, напри- клад, Еріан Верми в 2004 році описав приклад узагальненого MVC.

У передмові до дисертації «Naked objects» Річарда Поусона (Richard Pawson), - Трюгве Реенскауг згадує свою неопубліковану найбільш ранню версію MVC, згідно з якою :

- модель ставилася до «розуму» користувача;

- під поданням мався на увазі редактор, що дозволяє користувачеві переглядати і оновлювати інформацію;

- контролер був інструментом для зв'язування уявлень воєдино і застосовувався користувачем для вирішення його завдань.

Основна мета застосування цієї концепції полягає в відділенні бізнес-логіки (моделі) від її візуалізації (уявлення, виду). За рахунок такого поділу під- вищується можливість повторного використання коду.

Найбільш корисне застосування даної концепції в тих випадках, коли користувач повинен бачити ті ж самі дані одночасно в різних контекстах і / або з різних точок зору.

Зокрема, виконуються наступні завдання.

До однієї моделі можна приєднати кілька видів, при цьому не зачіпаючи реалізацію моделі. Наприклад, деякі дані можуть бути одночасно представлені у вигляді електронної таблиці, гістограми і кругової діаграми.

Не торкаючись реалізацію видів, можна змінити реакції на дії користувача (натискання мишею на кнопці, введення даних) - для цього досить використовувати інший контролер.

Ряд розробників спеціалізується тільки в одній з областей: або розробляють графічний інтерфейс, або розробляють бізнес-логіку. Тому можливо доби- тися того, що програмісти, які займаються розробкою бізнес-логіки (моделі), взагалі не будуть обізнані про те, яке уявлення буде використовуватися.

Концепція MVC дозволяє розділити модель, уявлення і контролер на три окремих компоненти.

Модель - надає дані і методи роботи з ними: запити до бази даних, перевірка на коректність. Модель не залежить від уявлення (не знає як дані візуалізувати) і контролера (не має точок взаємодії з користувачем) просто на- даючи доступ до даних і управління ними.

Модель будується таким чином, щоб відповідати на запити, змінюючи свій стан, при цьому може бути вбудовано повідомлення «спостерігачів».

Модель, за рахунок незалежності від візуального представлення, може мати кілька різних уявлень для однієї «моделі».

Представленнявідповідає за отримання необхідних даних з моделі і відправляє їх користувачеві. Представлення не обробляє жодних введених даних користувача.

Контролер забезпечує «зв'язок» між користувачем і систе- мою. Контролює і направляє дані від користувача до системи і навпаки. Використовує модель і уявлення для реалізації необхідного дії.

Функціональні можливості та розбіжності.

Оскільки MVC не має суворої реалізації, то реалізований він може бути по-різному. Немає загальноприйнятого визначення, де повинна розташовуватися бізнес-логіка. Вона може знаходитися як в контролері, так і в моделі. В останньому випадку, модель буде містити всі бізнес-об'єкти з усіма даними і функціями.

Деякі фреймворки жорстко задають де повинна розташовуватися бізнес-логіка, інші не мають таких правил.

Також не вказано, де повинна знаходитися перевірка введених користувачем даних. Проста валідація може зустрічатися навіть у поданні, але частіше вони зустрічаються в контролері чи моделлю.

Інтернаціоналізація і форматування даних також не має чітких вказівок по розташуванню.

Для реалізації схеми «Model-View-Controller» використовується досить велика кількість шаблонів проектування (в залежності від складності архітектурного рішення), основні з яких - «спостерігач», «стратегія», «компоновщик».

Найбільш типова реалізація - в якій уявлення відокремлено від моделі шляхом встановлення між ними протоколу взаємодії, що використовує «апарат подій» (позначення «подіями» певних ситуацій, що виникають в ході виконання програми, і розсилка повідомлень про них всім тим, хто підписався на отримання) : при кожному особливому зміні внутрішніх даних в моделі (позначеному як «подія»), вона сповіщає про нього, ті залежать від її

уявлення, які передплатили такого оповіщення - і уявлення оновлюється. Так використовується шаблон «спостерігач».

При обробці реакції користувача - уявлення вибирає, в залежності від реакції, потрібний контролер, який забезпечить той чи інший зв'язок з моделлю. Для цього використовується шаблон «стратегія», або замість цього може бути модифікація з використанням шаблону «команда».

Для можливості однотипного поводження з підоб'ектами складно-складеного ієрархічного виду - може використовуватися шаблон «компоновщик». Крім того, можуть використовуватися і інші шаблони проектування - наприклад, «фабричний метод», який дозволить задати за замовчуванням тип контролера для відповідного виду.

Ruby on Rails одним з перших почав використовувати схему поділу компонентів додатка Model-View-Controller (MVC), яка значно прискорює цикл розробки і дозволяє оперативно виправляти помилки, швидко реалізовувати бізнес-вимоги в проекті і вносити в них зміни.

Фреймворк пропагує «угоду поверх конфігурації». Це дає одноманітність структури і архітектурних принципів проектів, написаних з його використанням, що забезпечує збереження однакових підходів до розробки з проекту в проект. Якщо в команді трапиться ротація або прийдуть нові фахівці, однаковість підходу дозволить їм швидко влитися в проект, а бізнесу - скоротити час і знизити вартість підключення нових розробників.

Коли ведеться робота з базою даних, наприклад з Postgresql, Active Record (ORM, використовувана Ruby on Rails) дозволяє абстрагуватися від поточної БД на більш високих рівнях. Це дає можливість писати практично однаковий код під будь-яку підтримувану базу даних.

Фреймворк працює з усіма популярними SQL-базами даних, завдяки чому можна зосередитися на реалізації бізнес-задач, а не займатися написанням величезної кількості низькорівневих SQL-команд.

Ruby on Rails також підтримує шаблонізатор для веб-розробки, що прискорюють створення шаблонів для фронтенда. Самостійно писати HTML-код без допомоги автоматизованих генераторів складно, довго і дорого, тому фреймворк працює з безліччю шаблонізаторів:

- ERB;
- HAML;
- SLIM.

Деплой і тестування. Чим більше бізнес-логіки в додатку, тим складніше оновлювати його до актуальної версії. В Ruby є інструменти, що дають можливість провести оновлення однією командою, описавши на Ruby все стадії розгортання. Система автоматизації зробить все інше.

Щоб стежити за якістю коду, в екосистемі мови є інструменти, що дозволяють покрити тестами все області написання коду, починаючи від юніт-тестування і закінчуючи інтеграційним. З Ruby on Rails можна використовувати такі інструменти:

- Minitest;
- Rspec;
- Capybara;
- Cucumber.

Щомісяця і щороку для Ruby і фреймворка Rails виходять нові оновлення та виправлення. Зараз вже доступна шоста версія Ruby on Rails, в якій представлені:

-можливість паралельного тестування;

-підтримка роботи з декількома базами даних;

-інструменти для взаємодії з фронтенд-технологіями;

-вбудована система організації зберігання призначених для користувача файлів.

Велика частина змін і поліпшень пов'язана з тими вимогами, які висувають до мови розробники, вирішуючи ті чи інші завдання.

Наприклад, для зберігання призначених для користувача даних існує безліч інструментів, створених спільнотою. Розробники Ruby, розуміючи, що такі інструменти користуються попитом, реалізували необхідні можливості на рівні мови.

Технологія розвивається і силами спільноти. За статистикою GitHub (рис. 2.12), Rails має найбільше контрибуторів, які беруть активну участь у поліпшенні проекту, серед фреймворків інших мов.

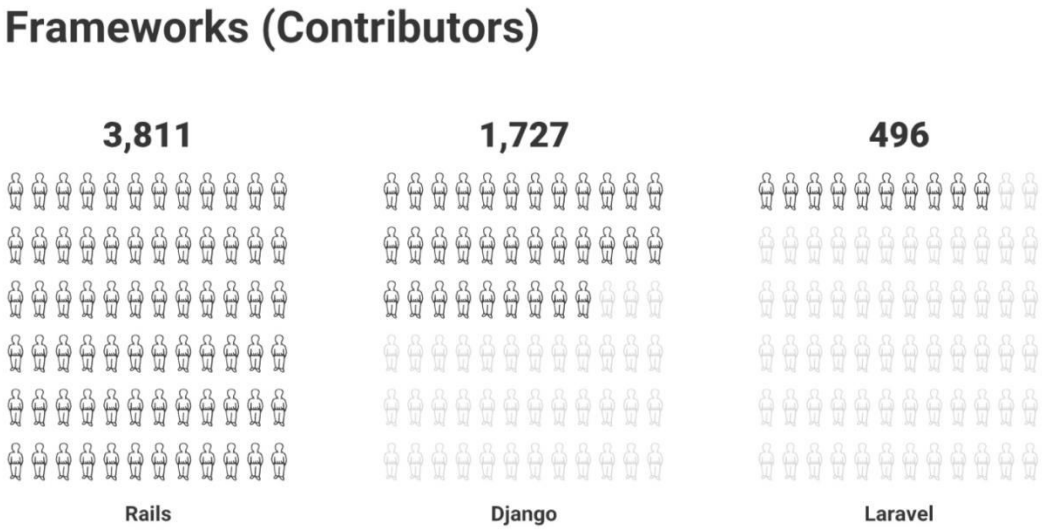

Рисунок 2.12 – статистика розвитку використання Ruby on Rails

Чим більше контрибуторів, тим вище ймовірність, що запит на поліпшення або на виправлення помилки буде реалізований швидше і це не доведеться робити всередині команди, витрачаючи час розробки проекту.

Об'єктно - орієнтоване програмування (ООП) - це модель програмування яка базується на стверджені того, що програма це сукупність об’єктів які взаємодіють між собою. Кожен об’єкт в цій моделі є незалежним, і він здатний отримувати, обробляти дані та відправляти ці дані іншим об’єктам. В ООП використано моделі успадкування, модульності, поліморфізму та інкапсуляції.

Основним поняттям ООП є об'єкт. Об'єкт можна визначити як певну сукупність даних(характеристик об'єкта) та методів роботи з ними. Для класифікації об'єктів у ООП використовують класи. Клас служить зразком для створення обєкту, тобто об'єкт є нічим іншим, ніж копією класу.

Кожен об'єкт має процедури і функції(те що він уміє виконувати, наприклад,завантажувати файл, відображати картинку і т.д.), які служать для роботи з даними об'єкта. Ці процедури і функції називаються методами.

Існування ООП можливе завдяки трьом основним парадигмам на яких базується саме ООП:

- Інкапсуляція. Також відома як приховування даних. Зміст інкапсуляції полягає у приховуванні від зовнішнього користувача деталей реалізації об'єкта, замість цього надаючи інтерфейс взаємодії з ним.

- Успадкування. Це означає, що об'єкти (класи) можуть переймати деякі властивості у своїх прабатьків. Як? Це залежить від тієї мови, на якому пишеться програма. Однак у будь-якому випадку картина та ж: це призводить до повторного використання вже написаного одного разу коду. Підкласи успадковують атрибути та поведінку своїх батьківських класів, і можуть мати нові власні атрибути. Тобто утворюється ієрархія з класів, де від основного класу(так званого, предка) походять усі інші класи.

- Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається (рис. 2.13), тобто, два або більше класів можуть реагувати по різному на однакові повідомлення. Це спричинене зміною в одного з класів якогось методу(процедури, функції), шляхом запису іншого алгоритму. Як приклад, деяка комп'ютерна програма при натисканні клавіші Esc завершить роботу, інша ж програма після натискання кнопки Esc тільки відкриє меню даної програми.

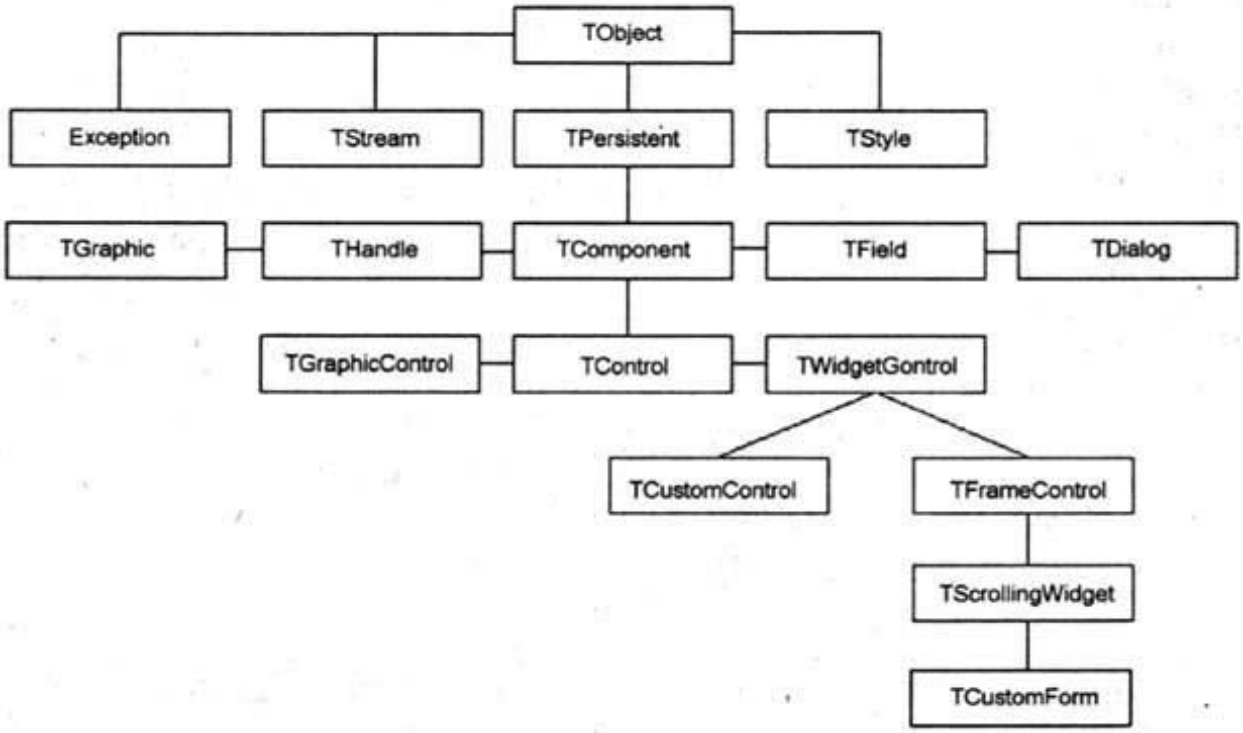

Рисунок 2.13 – Приклад ієрархії класів ООП

Висновки до розділу 2

В даному розділі було розглянуто проектні та технічні рішення, описані методи розробки та використання мови програмування, також її концепції. Розглянуто переваги та недоліки обраних методів розробки проекту та їхні відмінності.

Розглянуто схему поділу даних програми Model View Controller, її призначення, відмінності та актуальність використання. Описані функціональні можливості, концепції та реалізації технологій, HTML та CSS, їх використання, переваги та недоліки. Огляд особливостей фреймворку Bootstrap.

Розглянуті технології були вивчені та проаналізовані для подальшої роботи та застосуванням їх в проекті.

РОЗДІЛ 3 ОПИС РОЗРОБКИ ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЕКТУ

3.1 Встановлення Ruby та RoR

Встановлення Homebrew.

Спочатку нам потрібно встановити Homebrew (рис. 13). Homebrew дозволяє легко встановлювати та компілювати програмні пакети з джерела. Homebrew поставляється з дуже простим сценарієм встановлення. Також він попросить встановити інструменти XCode CommandLine.

Відкриваємо термінал і запускаємо команду:

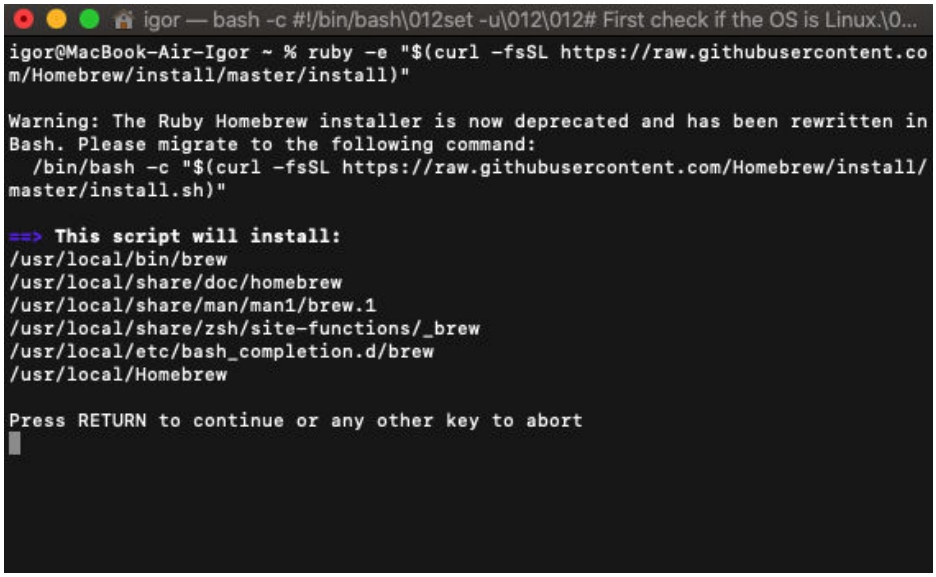

Рисунок 3.1 – Встановлення Homebrew

Далі встановлюєм Xcode CommandLine. Xcode (рис.13.1) - це великий набір інструментів розробки програмного забезпечення та бібліотек від Apple. Інструменти командного рядка Xcode є частиною XCode. Для встановлення багатьох поширених інструментів на основі Unix потрібен компілятор GCC. Інструменти командного рядка Xcode включають компілятор GCC.

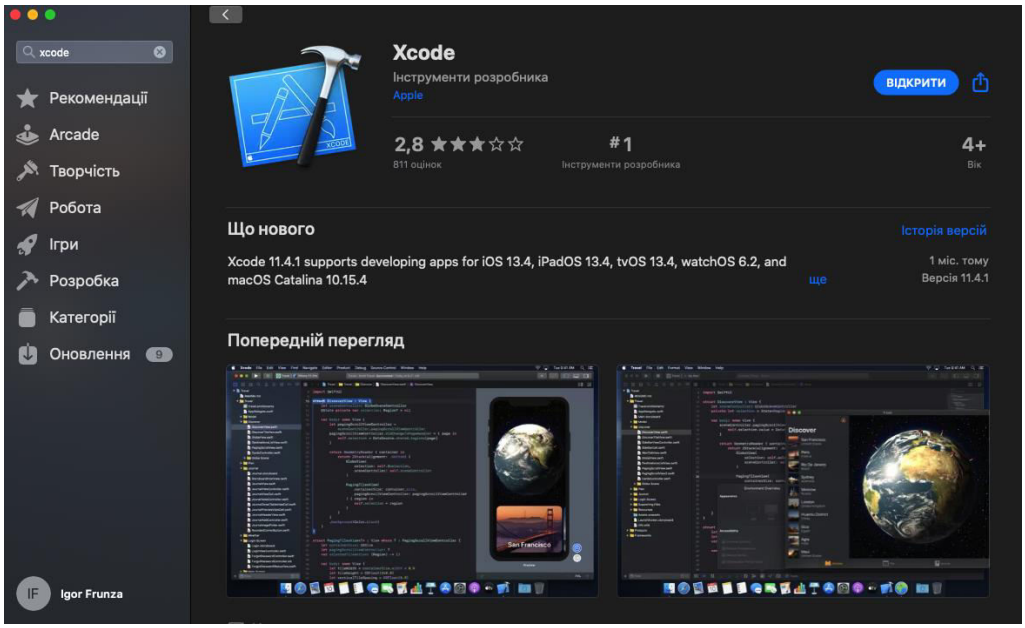

Рисунок 3.2 – Встановлення Xcode

Встановлення Ruby (рис. 13.2). Тепер, коли у нас встановлений Homebrew, ми можемо використовувати його для установки Ruby.

Ми будемо використовувати rbenv для встановлення та управління нашими версіями Ruby.Для цього запускаєм у своєму терміналі наступні команди:

```
brew install rbenv ruby-build

# Add rbenv to bash so that it loads every time you open a terminal
echo 'if which rbenv > /dev/null; then eval "$(rbenv init -)"; fi' >> ~/.zshrc
source ~/.zshrc

# Install Ruby
rbenv install 2.7.1
rbenv global 2.7.1
ruby -v
```

Рисунок 3.3 - Встановлення Ruby за допомогою Homebrew

Налаштування Git (рис. 14). Ми будемо використовувати Git для нашої системи контролю версій, тому ми будемо налаштовувати її відповідно до нашого облікового запису Github. Це стане в нагоді для майбутнього. Маючи розподілену архітектуру, Git є прикладом DVCS (отже, розподіленої системи управління версіями). Замість того, щоб мати лише одне місце для повної

історії версій програмного забезпечення, як це часто зустрічається в колись популярних системах контролю версій, таких як CVS або Subversion (також відомий як SVN), у Git робоча копія коду кожного розробника є також сховищем. які можуть містити повну історію всіх змін.

Окрім розповсюдження, Git був розроблений з врахуванням продуктивності, безпеки та гнучкості.

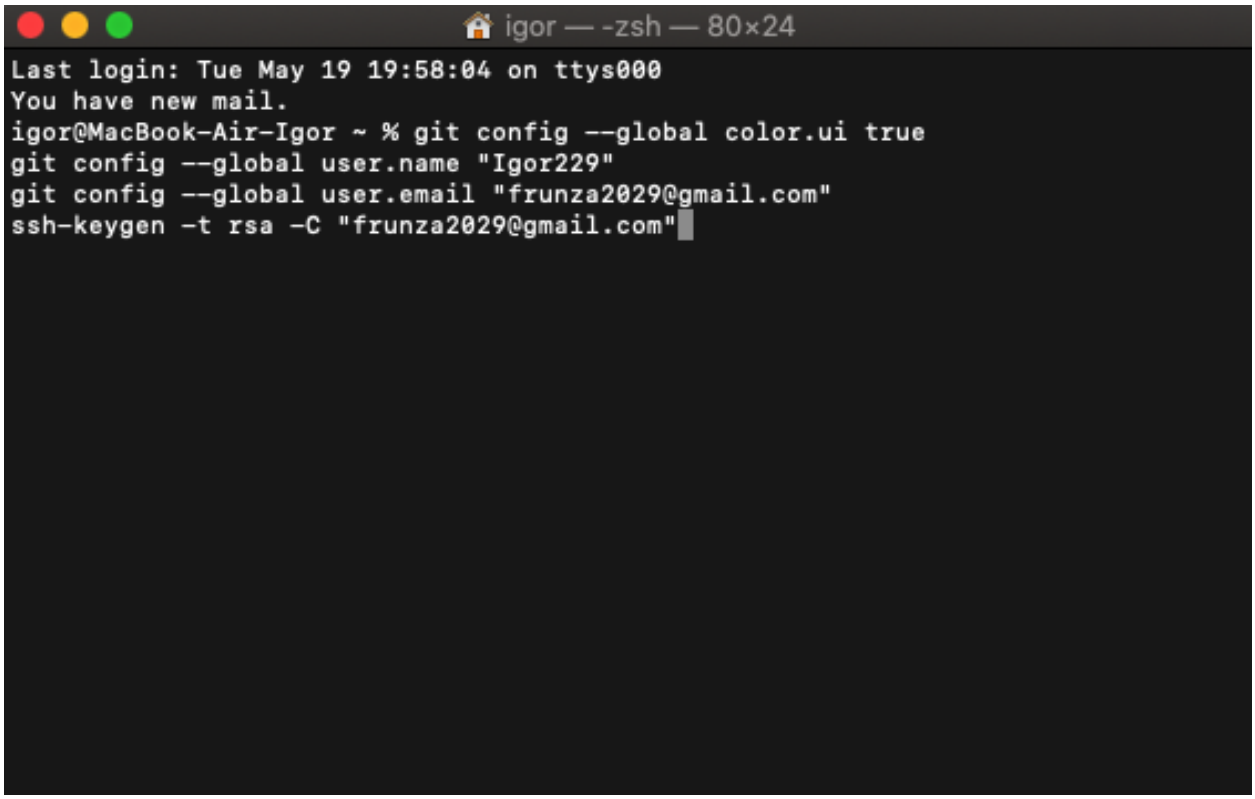

Рисунок 3.4 - Налаштування Git

Наступний крок - взяти щойно створений ключ SSH та додати його до свого облікового запису Github (рис. 3.4).

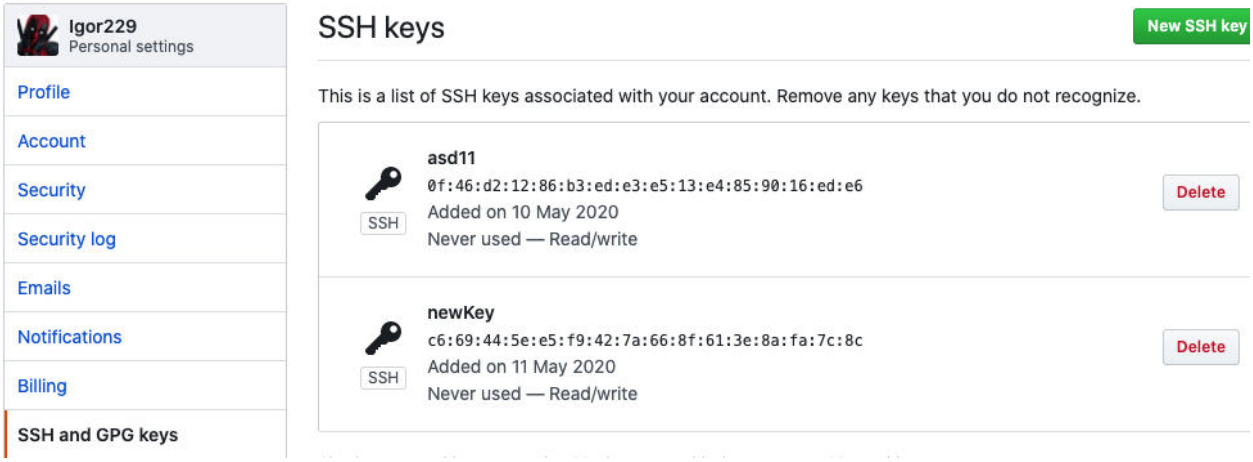

Рисунок 3.5 - Зберігання SSH ключа Git

Встановлення Ruby on Rails (рис.3.5). Встановити Rails просто, виконуєм таку команду у своєму терміналі:

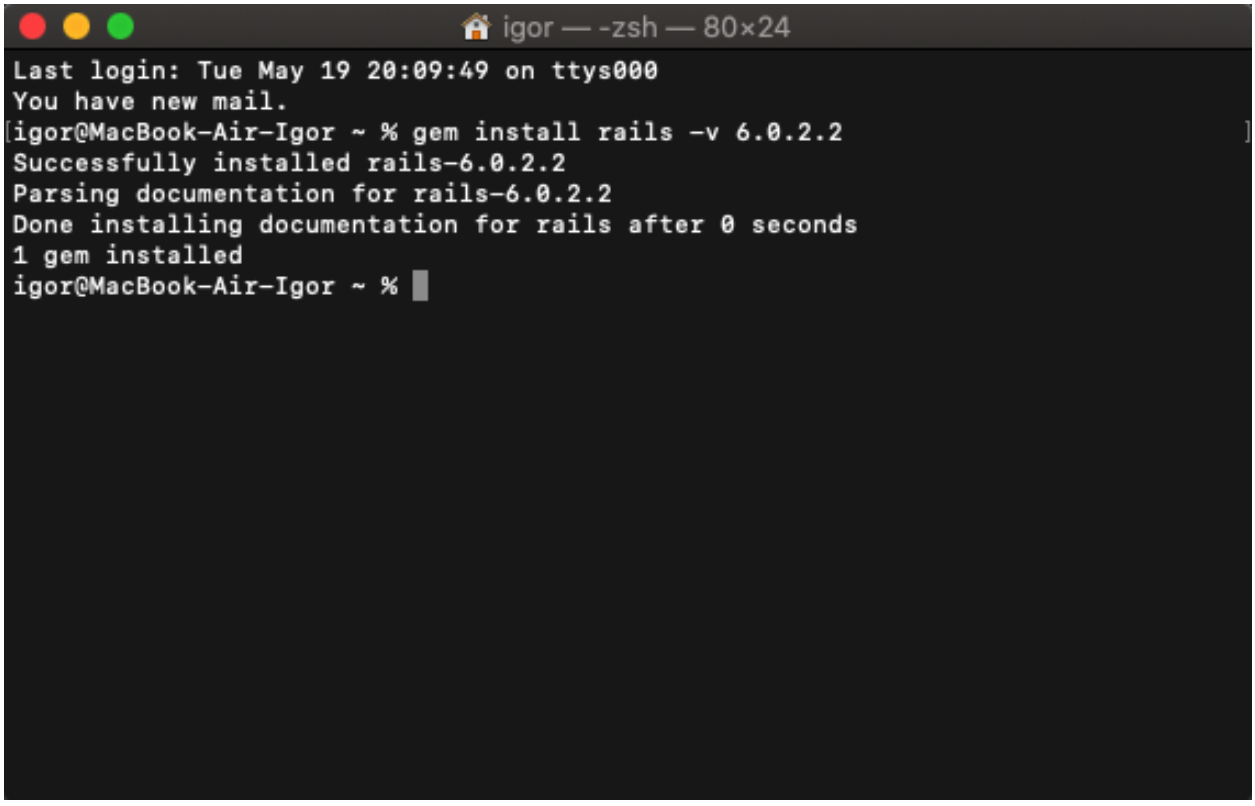

Рисунок 3.6 - Встановлення фреймворку Ruby on Rails

Rails встановлені, але для того, щоб ми використовували його, нам потрібно сказати rbenv, щоб той його побачив командою rbenv rehash.

Налаштування бази даних. Потрібно встановити Sqlite3 за допомогою Homebrew (рис. 3.6, 3.7), тому що при використанні вбудованої версії з macOS, є велика ймовірність стикнутись з деякими проблемами.

SQLite - це бібліотека на мові C, яка реалізує невелику, швидку, автономну, високонадійну, повнофункціональну систему баз даних SQL. SQLite - це найбільш використовуваний двигун бази даних у світі. SQLite вбудований у всі мобільні телефони та більшість комп'ютерів і постачається у безлічі інших програм, якими користуються люди щодня.

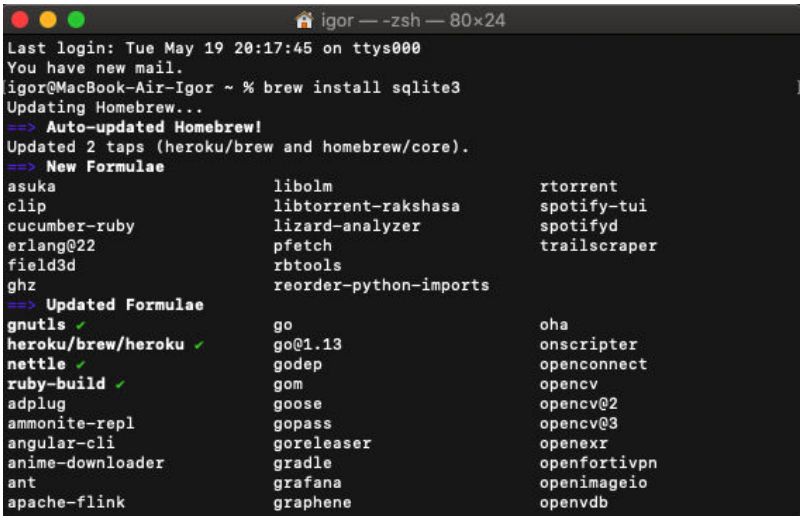

Рисунок 3.6 - Встановлення бази даних Sqlite3

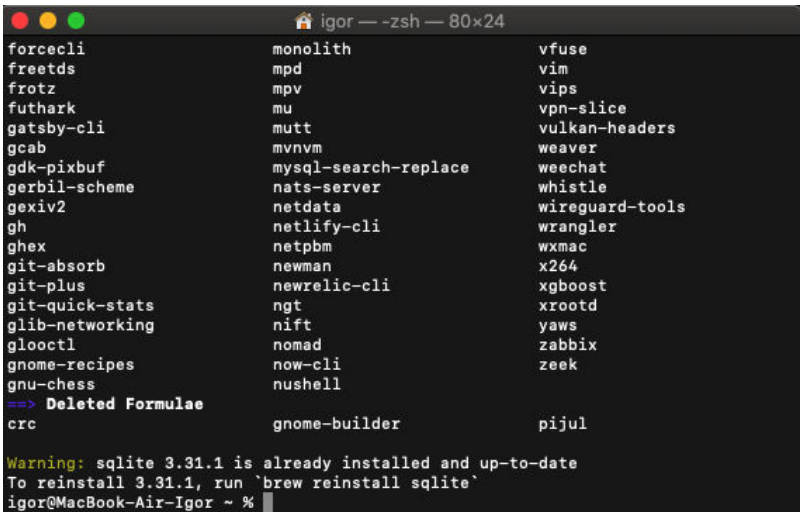

Рисунок 3.7 - Додаток до рисунку 15

Mojave (одна з версій ОС mac) змінив розташування файлів заголовків, необхідних для компіляції розширень C. Потрібно виконати таку команду, щоб встановити pkg, nokogiri (рис. 3.8) або інші gem файли, для яких потрібні розширення C:

```
sudo installer -pkg /Library/Developer/CommandLineTools/Packages/macOS_SDK_headers_for_macOS_10.14.pkg
-target /
```

Рисунок 3.8 - встановлення пакету nokogiri

3.2 Розробка проекту з детальним описом дій

Створюєм проект на Rails з ім'ям volunteer_app в директорії vapp (рис. 17). Rails автоматично встановлює геми, залежно які задані в Gemfile при використанні bundle install.

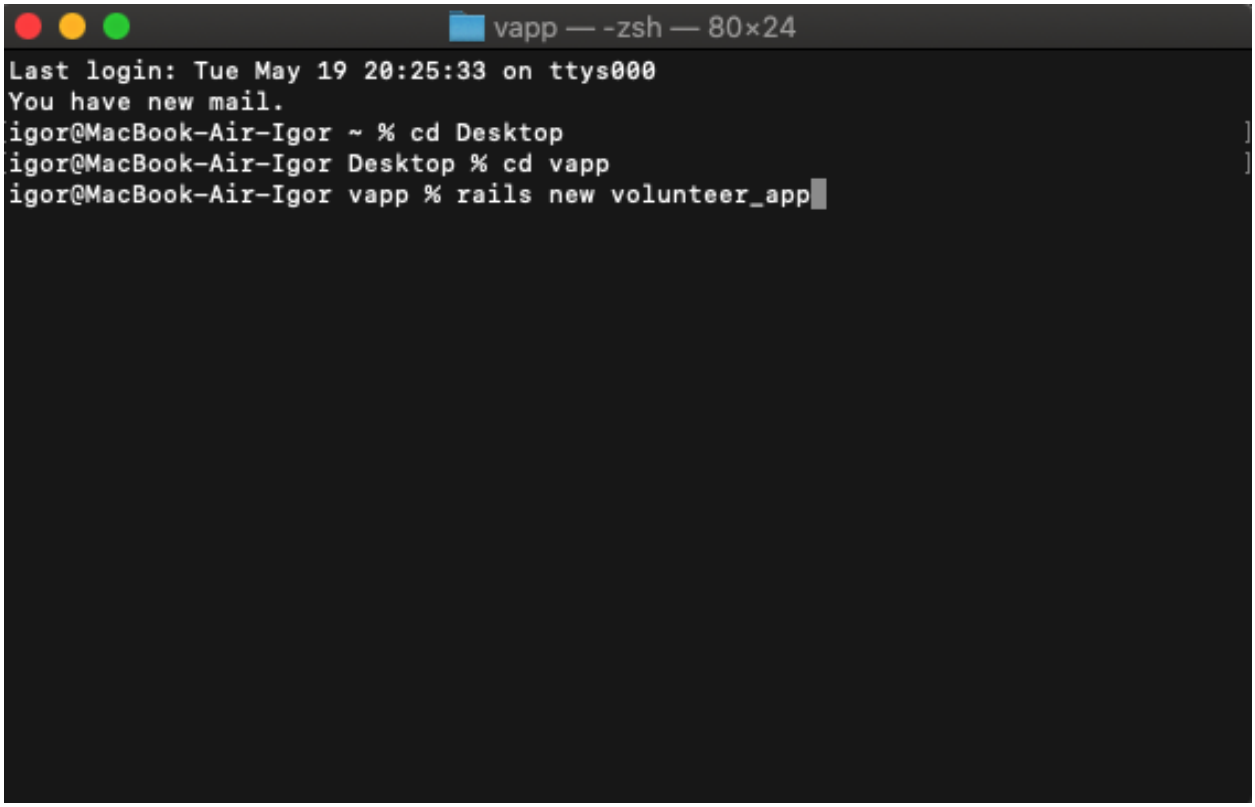

Рисунок 3.9 - Створення проекту Ruby on Rails

Після виконання попередньої команди, в директорії volunteer_app вже згенеровані необхідні елементи проекту (рис. 3.10).

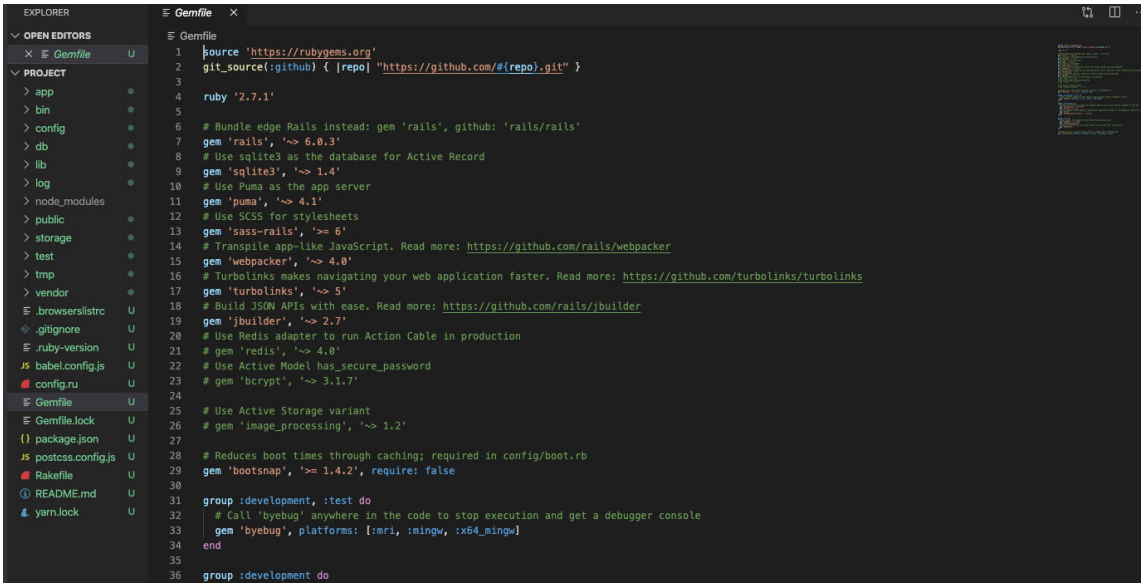

Рисунок 3.10 - Gem файли створеного проекту

Далі додаєм необхідні gem файли та встановлюєм їх за допомогою відповідної команди (рис. 3.11).

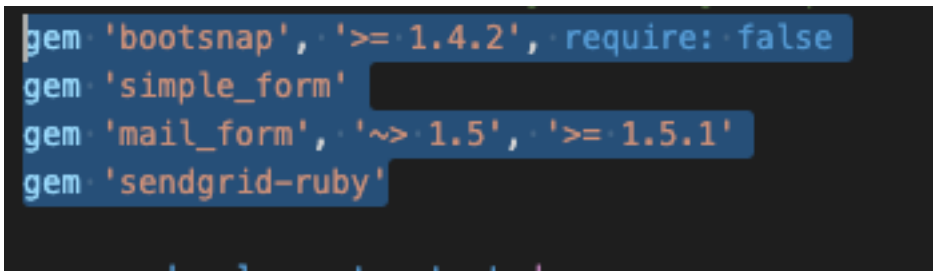

Рисунок 3.11 - Додавання необхідних gem файлів

Насамперед потрібно згенерувати контроллер (рис. 3.12) через відповідну команду в терміналі та налаштувати роути, для створення посилань в браузері та корректної роботи веб-сайту (рис. 3.13).

```
igor@MacBook-Air-Igor project % rails g controller contacts
warning ../package.json: No license field
Running via Spring preloader in process 40798
      create  app/controllers/contacts_controller.rb
      invoke  erb
      create    app/views/contacts
      invoke  test_unit
      create    test/controllers/contacts_controller_test.rb
      invoke  helper
      create    app/helpers/contacts_helper.rb
      invoke    test_unit
      invoke  assets
      invoke    scss
      create      app/assets/stylesheets/contacts.scss
igor@MacBook-Air-Igor project %
```

Рисунок 3.12 - Генерування controller contact

```
config > routes.rb
1   Rails.application.routes.draw do
2     root to: 'main#index'
3     resources :contacts, only: [:new, :create]
4     # For details on the DSL available within thi
5   end
```

Рисунок 3.13 - Налаштування роутів

Щоб створити форму реєстрації, потрібно налаштувати контроллер, створивши в ньому відповідний метод (рис. 3.14).

```
class ContactsController < ApplicationController
    def new
        @contact = Contact.new
    end

    def create
        @contact = Contact.new(params[:contact])
        @contact.request = request
        if @contact.deliver
            flash.now[:error] = nil
        else
            flash.now[:error] = 'Cannot send message.'
            render :new
        end
    end
end
```

Рисунок 3.14 - Метод створення форми анкети для реєстрації

Наступний крок - створення моделі та відповідні валідації (рис. 3.15), а також налаштування відправлення заповненої форми на вказану пошту, через яку можна буде зв’язатися з користувачем.

app > models > contact.rb

```ruby
class Contact < MailForm::Base
    attribute :name,        :validates => true
    attribute :email,       :validates => /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i
    attribute :message,     :validates => true
    attribute :nickname,    :validates => true
    attribute :phone,       :validates => true
    attribute :date,        :validates => true
    attribute :city,        :validates => true

    def headers
      {
        :subject => "Contact Form",
        :to => "frunza2029@gmail.com",
        :from => %("#{name}" <#{email}>)

      }
    end
end
```

Рисунок 3.15 – валідація та налаштування моделі contact.rb

Створивши контроллер, модель та виконавши всі необхідні налаштування, можна почати створювати Вид нашої сторінки. За допомогою HTML та CSS, створюєм header «шапку» сайту з елементами навігації та логом сайту (рис. 3.16). Далі створюєм форму анкети реєстрації (рис. 3.17) вже за допомогою синтаксису Ruby, також задаєм стилі по класах.

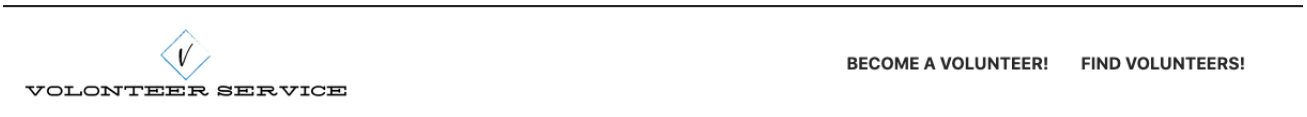

Рисунок 3.16 - Вигляд шапки сайту

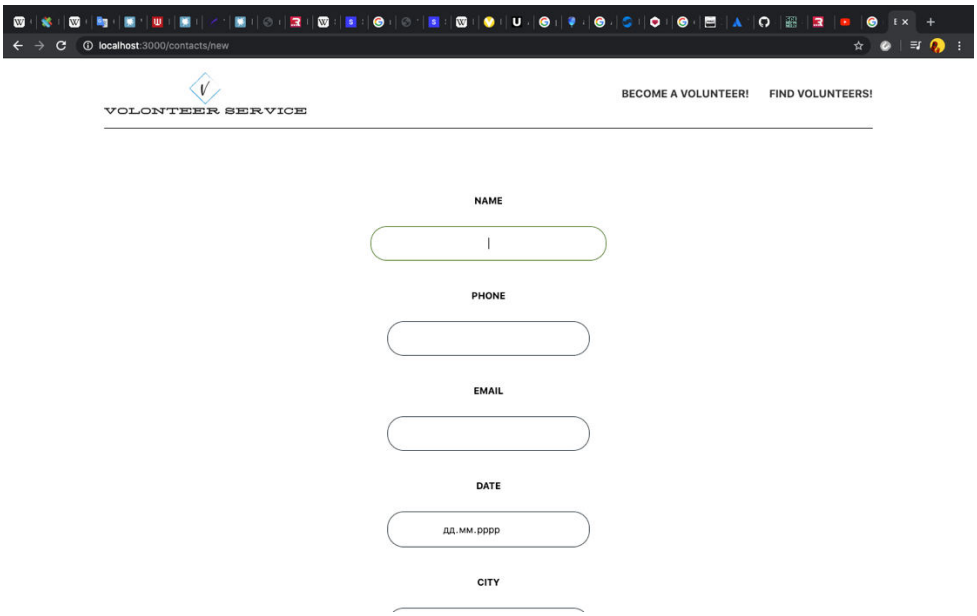

Рисунок 3.17 - Вигляд анкети реєстрації

Після заповнення анкети (рис. 3.19), користувача пересилає на сторінку з повідомленням про успішно надіслану форму реєстрації (рис. 3.20). Всі поля в анкеті є обов'язковими, у разі якщо користувач не заповнив одне з полів, з'явиться відповідне повідомлення. (рис. 3.18) Далі натиснувши на лого сайту можна повернутись на головну сторінку.

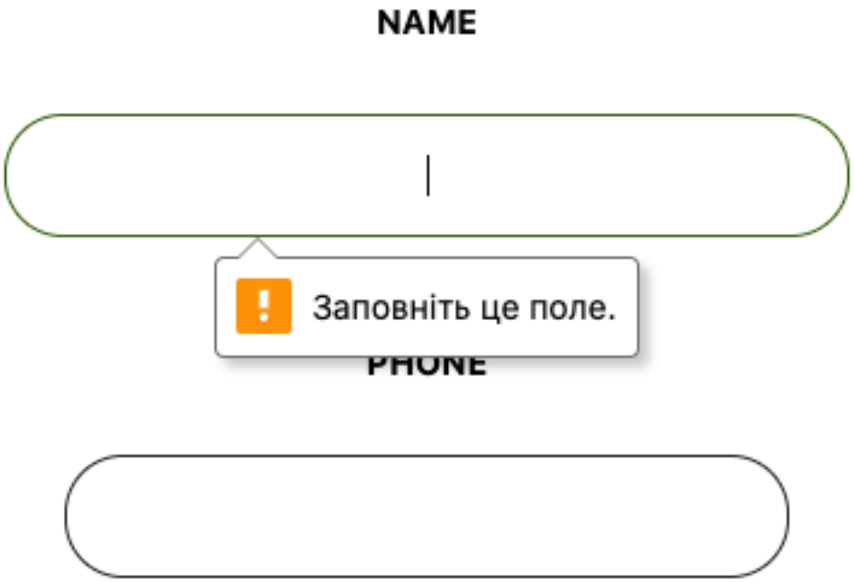

Рисунок 3.18 - Валідація анкети реєстраці

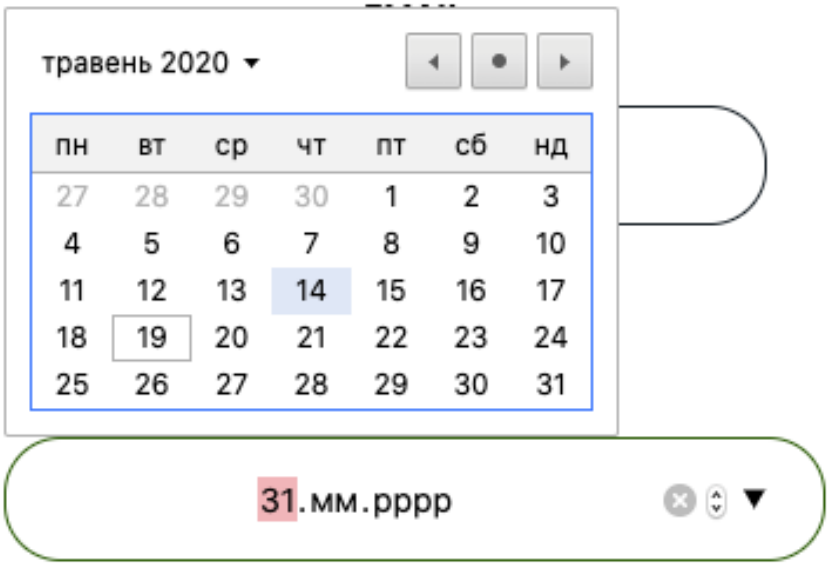

Рисунок 3.19 - Вибір дати народження

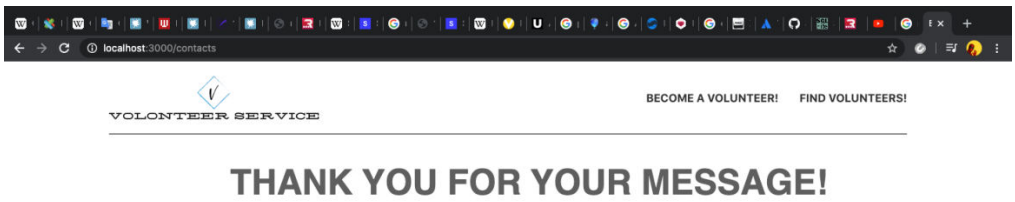

Рисунок 3.20 - Пересилання після відправлення форми

Зберігання проекту в GitHub (рис. 3.21). Git дозволяє повертати окремі файли і весь проект до попереднього стану, переглядати відбуваються з часом зміни. Визначати, хто останнім вносив зміни у раптово зупинивший роботу модуль, відстежити ланцюг подій, яка привела до помилок і багато

іншого. Одним з найбільш популярних ресурсів для роботи з Git є GitHub. Для зручності роботи з ним існує графічний клієнт GitHub Desktop і консольний Git Shell.

За допомогою вкладки commits можна вивчити історію розробки проекту: коли і ким були внесені зміни в які файли і рядки, як розвивався проект в цілому.

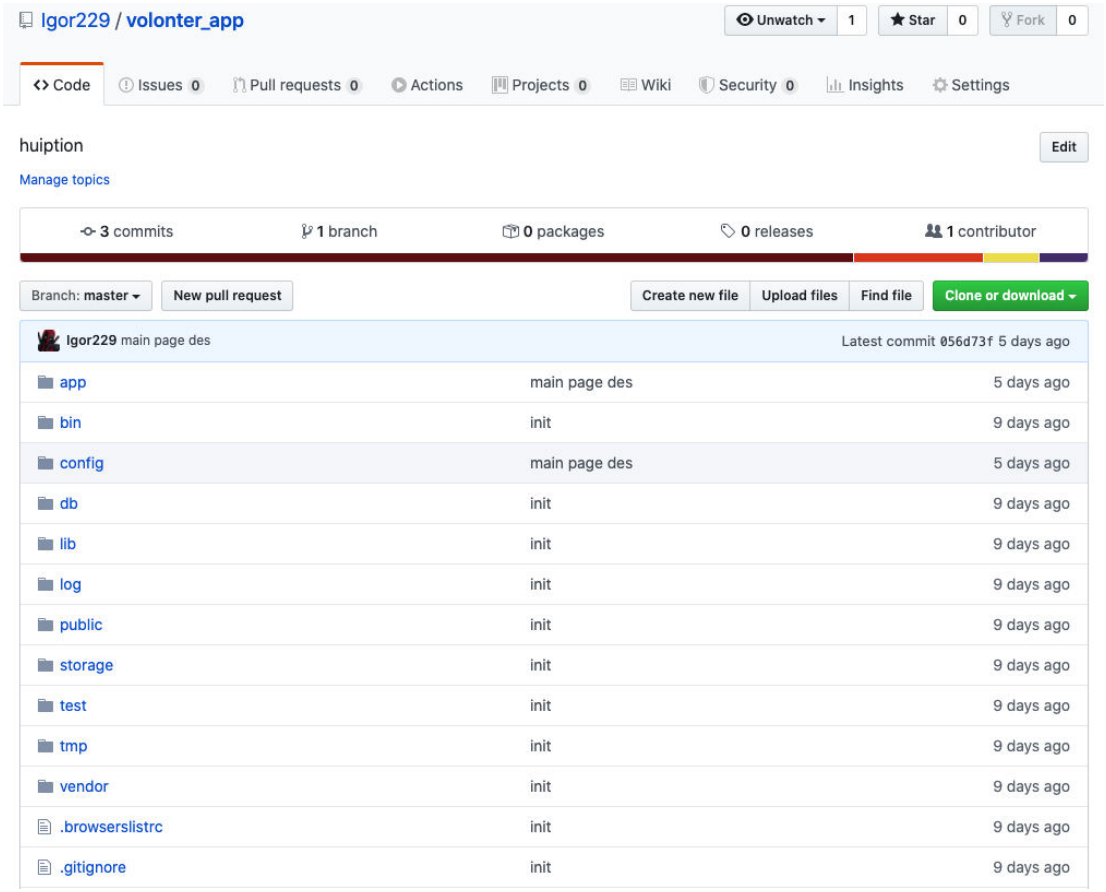

Рисунок 3.21 - Репозиторій GitHub

Heroku - хмарна PaaS-платформа, що підтримує ряд мов програмування. Компанією Heroku володіє Salesforce.com. Heroku, одна з перших хмарних платформ, з'явилась в червні 2007 року і спочатку підтримувала тільки мову програмування Ruby, але на даний момент список підтримуваних мов також включає в себе Java, Node.js, Scala, Clojure, Python і

PHP. На серверах Heroku використовуються операційні системи Debian або Ubuntu (яка також заснована на Debian).

SendGrid - це платформа, яка поліпшує транзакційні розсилки і дозволяє масштабувати їх для вирішення задач email-маркетингу. Вона пропонує гнучкі веб та SMTP API-інтерфейси, може легко інтегруватися з будь-якою хмарною інфраструктурою. SendGrid підтримує різні фреймворки, мови і додатки.Основні характеристики SendGrid:

- SMTP-серві;
- Кастомниє інтеграції API;
- Відстеження відкриттів і кліків;
- Шаблони повідомлень;
- Відмова від відстеження;
- Моніторинг репутації;
- Управління списками;
- Виділені IP-адреси;
- SMTP API;
- Моніторинг ISP;
- Тестування фільтра;
- Балансування навантаження;
- Аналіз Webhook;
- Зворотній зв'язок;
- DKIM Налаштування;
- SMTP Relay.

Так як в нас встановлений Homebrew, використовуєм його для того щоб встановити Heroku CLI (рис. 3.22). Після встановлення, Heroku commands доступні в нашому терміналі.

```
brew tap heroku/brew && brew install heroku
```

Рисунок 3.22 - Встановлення Heroku CLI

SendGrid можна приєднати до програми Heroku через CLI (рис. 3.22).

```
$ heroku addons:create sendgrid:starter
-----> Adding sendgrid:starter to sharp-mountain-4005... done, v18 (free)
```

Рисунок 3.22 - Приєднання SendGrid

SendGrid має gem Ruby, який сприяє прийняттю SendGrid у додатках Ruby. Залишилось налаштувати ActionMailer для використання SendGrid (рис. 3.23). В директорії config відкриваєм файл environment.rb і вводим налаштування ActionMailer, щоб вказати на сервери SendGrid.

```
ActionMailer::Base.smtp_settings = {
  :user_name => ENV['SENDGRID_USERNAME'],
  :password => ENV['SENDGRID_PASSWORD'],
  :domain => 'yourdomain.com',
  :address => 'smtp.sendgrid.net',
  :port => 587,
  :authentication => :plain,
  :enable_starttls_auto => true
}
```

Рисунок 3.23 - Налаштування ActionMailer SendGrid

3.3Розробка frontend частини сайту

Стилізація сайту реалізувалась за допомогою HTML та CSS.

Зазвичай HTML-документ - це файл з розширенням .html або. htm, в якому текст розмічений HTML-тегами (рис. 3.24). Засобами HTML задаються синтаксис і розміщення тегів, відповідно до яких браузер відображає вміст Веб-документа. Текст самих тегів Веб-браузером не відображається.

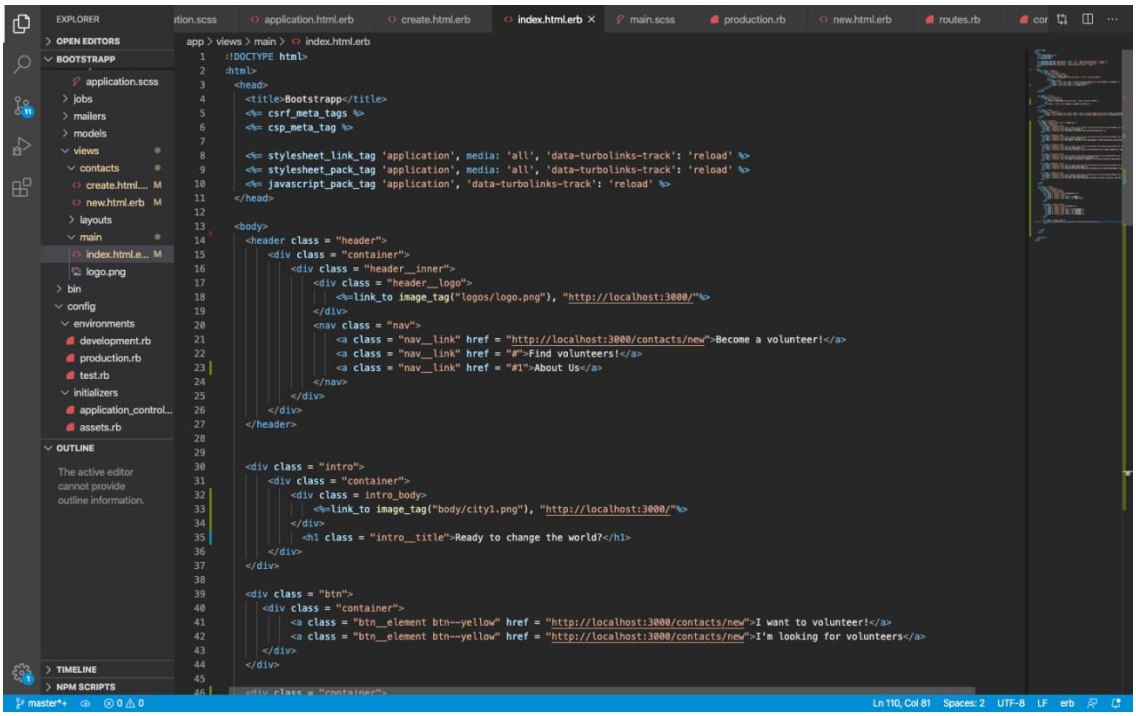

Рисунок 3.24 - Вміст файлу index.html.erb

HTML (HyperText Markup Language) говорить браузеру, який зміст сторінки, наприклад, «заголовок», «параграф», «список», «елемент списку». CSS (Cascading Style Sheets) говорить браузеру, як відображати елементи.

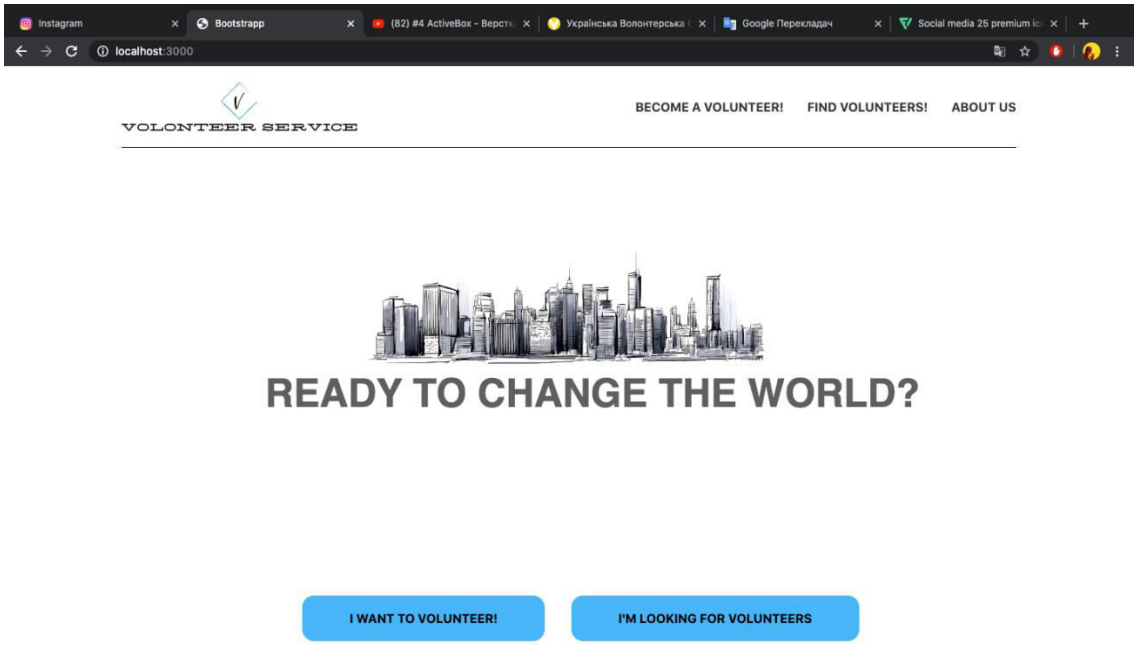

Рисунок 3.25 - Вигляд головної сторінки сайту

Головною метою стилізації сайту було створення приємного та простого у використанні інтерфейсу сайту (рис. 3.25, 3.26).

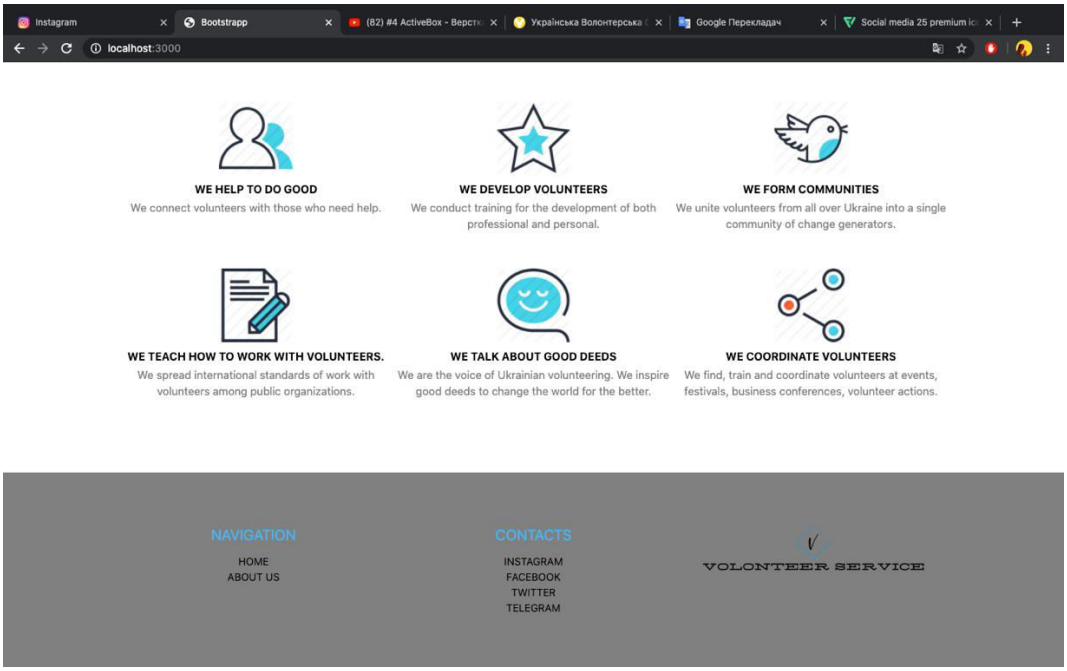

Рисунок 3.26 - Додаток до рисунку 3.25

Висновки до розділу 3

В даному розділі описаний процес створення онлайн сервісу реєстрації волонтерів. Зображено підхід до вирішення певних завдань, встановлення необхідних програм та ресурсів, налаштування форм та стилізація сайту.

Використано такі технології як:

- фреймворк Ruby on Rails
- система контролю версій Github
- HTML, CSS

Розроблено форму реєстрації за допомогою якої, користувач надсилає основні дані для подальшого зв’язку з ним. Також створено сторінку, яка описує чим саме сервіс допомагає людині, та які задачі вирішує. В хедері розміщено логотип та три кнопки:

- стати волонтером
- знайти волонтерів
- About us

Знизу створено футер, в якому присутні навігація по сайту, контакти та логотип.

ВИСНОВКИ

В даному проекті було розглянуто існуючі рішення та методи розробки проекту, переваги вибраних технологій. Було поставлено ряд головних задач, які виконувались в процесі розробки. Створено use case діаграму, розглянуто діаграми класів сервісу, gem-файли, які допомагають полегшити створення реєстрації. Описано процес встановлення необхідних технологій, процес розробки самого веб-сервісу онлайн реєстрації волонтерів для інформаційно-розважальних подій. Також проведений опис написання стилів, frontend частини сайту, за допомогою якого створено зручний у використанні інтерфейс для користувача.

Проведений аналіз актуальності теми волонтерства, її недоліки, переваги та користь яку надає.

У даній роботі було здійснено розробку онлайн сервісу реєстрації волонтерів. Розроблений сайт є досить актуальним в наш час, адже поширення волонтерського руху в Україні, утворення волонтерства для кожного - важлива складова суспільства, можливість допомогти українцям брати участь у волонтерських проектах.

Для розробки даного проекту було вибрано сучасні технології створення веб-сайтів, основним з яких є фреймворк Ruby on Rails.

Основною перевагою мови програмування Ruby і фреймворка Ruby on Rails вважається швидкість розробки, що стало хорошим вибором для створення сайту. Інстальовані та вставлені всі необхідні бібліотеки, та модулі, встановлено Homebrew. Створена форма реєстрації, валідація, стилізація за допомогою HTML, CSS. Сайт «задеплоїно» на хмарну платформу Heroku.

Оскільки в багатьох людей волонтерство асоціюється тільки з безкоштовною працею, то така пропозиція не надто приваблює. А якщо подивитись на це з іншої сторони, то для людей це можливість безкоштовно жити в іншій країні, отримувати досвід роботи в різноманітних галузях, допомагаючи людям, покращуючи екологію.

У роботі поставлена мета створити онлайн сервіс який б, допоміг людям стати волонтером та віднайти щось нове або здобути нові навички для себе, або ж знайти волонтерів для організації власної події.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ethan CeramiWeb Services EssentialsDistributed Applications with XMLRPC, SOAP, UDDI & WSDL. – Publisher: O'Reilly Media, February 2002. – 308p.

2. Брайан ТрэвисНазвание: XML и SOAP программирование для серверов. –

М.: Русская Редакция, 2001. – 216с.

3. Ken Pugh Interface Oriented DesignWith Patterns. / Publisher: Pragmatic Bookshelf, July 2006. – 240p.

4. Debbie StoneUser Interface Design and Evaluation / Debbie Stone, Debbie Stone, Caroline Jarrett, Mark Woodroffe, ShaileyMinocha. – Publisher: Elsevier / Morgan Kaufmann, April 2005. – 704p.

51

5. Eric Elliott Programming JavaScript ApplicationsRobust Web Architecture

With Node, HTML5, and Modern JS Libraries. / Publisher: O'Reilly Media, February 2013. – 300p.

6. Мухин В.И. Основы теории управления / В.И. Мухин. – М.:Экзамен, 2002. – 256с.

7. Вертакова Ю.В, Козьева И.А., Кузьбожев Э.Н. Управленческие решения: разработка и выбор / Ю.В.Вертакова, И.А.Козьева, Э.Н. Кузьбожев– М.: КНОРУС, 2005 – 351с.

8. Литвак Б.Г. Экспертные технологии в управлении. – М.: Дело, 2004.

263с.

9. Платов В.Я. Современные управленческие технологии. – М.: Дело, 2006.

168с.

10. Bloch, J. Effective Java Programming Language Guide,Addison-Wesley,

Boston, MA, 2001.– 462p.

11. Clarke, S. ―API Usability and the Cognitive Dimensions Framework‖, 2003. –

217p.

12. Ellis, B., Stylos, J. and Myers, B. The Factory Pattern in API Design: A

Usability Evaluation. International Conference on Software Engineering, 2007.

REST.aspx

13. Webservices[Електронний ресурс] – Режим доступа: https://www.eeducation.psu.edu/geog583/files/geog583/keynote_chappell.pdf

14. Definision of SOAP and REST[Електронний ресурс] – Режим доступа:

http://spf13.com/post/soap-vs-rest

15. Web Services, Part 1: SOAP vs. REST [Електронний ресурс] –

Режимдоступа:http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest

16. Википедия [Електронний ресурс] – Режим доступа:

https://ru.wikipedia.org/

17. Прикладной программный интерфейс[Електронний ресурс] –

Режим доступа: http://www.osp.ru/cw/2000/09/3539/

18. A Web application as a REST API client [Електронний ресурс] –

Режимдоступа: http://supervisord.org/api.html

19. JavaCC [Електронний ресурс] – Режим доступа:

http://www.engr.mun.ca/~theo/JavaCC-Tutorial/

20. Javacc-maven-plugin[Електронний ресурс] – Режим доступа:

http://mojo.codehaus.org/javacc-maven-plugin/

21. ConfluenceXML-RPCandSOAPAPIsplugin[Електронний ресурс] – Режим

доступа: https://developer.atlassian.com/display/CONFDEV/

22. SOAP-vs-REST [Електронний ресурс] – Режим доступа:
http://www.alliancetek.com/Blog/post/2012/10/12/SOAP-vs-REST.aspx
23. Webservices[Електронний ресурс] – Режим доступа: https://www.eeducation.psu.edu/geog583/files/geog583/keynote_chappell.pdf
24. Definision of SOAP and REST[Електронний ресурс] – Режим доступа:
http://spf13.com/post/soap-vs-rest
25. Web Services, Part 1: SOAP vs. REST [Електронний ресурс] –
Режимдоступа:http://www.ajaxonomy.com/2008/xml/web-services-part-1

КВАЛІФІКАЦІЙНА РОБОТА

КР.ІПз – 18.00.000 ПЗ

Група ІПз-2016

Фрунза І.П.

2020

ПВНЗ Університет Короля Данила

Кафедра Інформаційних технологій та програмної інженерії

УДК 004.415

КВАЛІФІКАЦІЙНА РОБОТА

Тема **Розробка онлайн сервісу реєстрації волонтерів для інформаційно-розважальних подій**

Спеціальність ***121 «Програмна інженерія»***

(код і назва спеціальності)

ПОЯСНЮВАЛЬНА ЗАПИСКА

КР.ПІз – 18.00.000 ПЗ

(позначення)

Студент

Фрунза І.П.

(підпис) (дата) (розшифрування підпису)

Керівник проекту

к.ф.-м.н. Остафійчук П.Г.

(посада) (підпис) (дата) (розшифрування підпису)

Допускається до захисту

Завідувач кафедри

д.т.н.,доц. Мануляк І.З.

(посада) (підпис) (дата) (розшифрування підпису)

2020

ПВНЗ УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет *Інформаційних технологій*
Кафедра *Інформаційних технологій та програмної інженерії*
Спеціальність *121 «Програмна інженерія»*

ЗАТВЕРДЖУЮ:
Завідувач кафедри ІТПІ
_____ *І.З.Мануляк*
"____" _____ 2019 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студенту *Фрунза Ігорю Петровичу*

1. Тема проекту (роботи) *Розробка онлайн сервісу реєстрації волонтерів для інформаційно-розважальних подій*

Затверджена наказом ректора Університету Короля Данила від *08.11.2019 р. № 22/4 – НВ*
2. Термін здачі студентом закінченого проекту (роботи) *11.06.2020 р.*
3. Вихідні дані до проекту (роботи) *Ruby, Ruby on Rails,*

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити) *1.Розглянути методи розробки онлайн сервісу. 2. Розробка форми реєстрації. 3. Реалізація структури веб-сайту.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Мета, об'єкт, завдання роботи. Аналіз існуючих систем реєстрації. Аналіз розробки проекту. Результати роботи. Висновки.

6. Консультанти з проекту (роботи), із зазначенням розділів проекту, що стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 11.11.2019

Керівник _____Остафійчук П.Г.

Завдання прийняв до виконання _____Фрунза І.П.

КАЛЕНДАРНИЙ ПЛАН

Пор №	Назва етапів дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1.	Отримання завдання та огляд аналогічних моделей	02.10.2019	Виконано
2.	Створення вимог до проекту	12.11.2019	Виконано
3.	Розробка моделі сервісу	14.12.2019	Виконано
4.	Створення форми реєстрації	23.12.2019	Виконано
5.	Встановлення Heroku	09.01.2020	Виконано
6.	Налаштування SendGrid	27.01.2020	Виконано
7.	Робота з Action Mailer	23.02.2020	Виконано
8.	Стилізація сайту	07.03.2020	Виконано
9.	Тестування	14.03.2020	Виконано
10.	Опис документації	15.04.2020	Виконано

Студент-дипломник _____ *Фрунза І.П.*
(підпис) (розшифровка підпису)
Керівник проекту _____ *Остафійчук П. Г.*
(підпис) (розшифровка підпису)

АНОТАЦІЯ

В роботі розкрита тема волонтерства, формування волонтерських спільнот та поширення їх за допомогою онлайн сервісу, наведені приклади існуючих аналогів. Проаналізовано діяльність волонтерських рухів та їх користь для суспільства. Розглянуто сучасні технології та їх методи вирішення поставленої задачі. Для реалізації онлайн сервісу реєстрації волонтерів, було обрано сучачні технології розробки веб-сайтів, такі як: Ruby on Rails, Sqlite3, Heroku, SendGrid. У роботі за результатами виконаних теоритичних та практичних досліджень розроблено онлайн сервіс, описано процес розробки форм реєстрації та стилізації сайту.

Ключові слова: онлайн сервіс, MVC, Ruby, Ruby on Rails, Sqlite, волонтер, веб-сайт, події.

SUMARRY

The paper reveals the topic of volunteering, the formation of volunteer communities and their dissemination through an online service, examples of existing analogues. The activity of volunteer movements and their benefit for society are analyzed. Modern technologies and their methods of solving the problem are considered. To implement the online volunteer registration service, modern web development technologies were chosen, such as: Ruby on Rails, Sqlite3, Heroku, SendGrid. In the work on the results of theoretical and practical research developed an online service, describes the process of developing forms of registration and stylization of the site.

Keywords: online service, MVC, Ruby, Ruby on Rails, Sqlite, volunteer, website, events.