



Agosto 27, 2023

Proyecto de Lenguajes de Programación II

Escuela Superior Politécnica del Litoral (ESPOL) - PAO I 2023

Juan Antonio González

juangonz@espol.edu.ec

ESPOL (FIEC)

Paul Gudiño

pgudino@espol.edu.ec

ESPOL (FIEC)

Cristopher Villa

cgvilla@espol.edu.ec

ESPOL (FIEC)

Índice

1 Características del Proyecto	4
1.1 Categoría	4
1.2 Temática y Descripción	4
1.3 Alcance	4
1.3.1 Historias de Usuario	4
1.4 Lenguajes de Programación y Paradigmas	5
1.4.1 Backend	5
1.4.2 Frontend	5
2 Protototipo	7
2.1 Prototipo de Baja Fidelidad en Figma	7
3 Gestión del Proyecto	9
4 Desarrollo	10
4.1 Backend	10
4.1.1 Endpoint: GET /api/ciudad	10
5 Anexos	11
5.1 Código Fuente	11
Bibliografía	12

Lista de Figuras

Figura 1: Prototipo de Baja Fidelidad - Pantalla 1	7
Figura 2: Prototipo de Baja Fidelidad - Pantalla 2	8
Figura 3: Prototipo de Baja Fidelidad - Pantalla 3	8
Figura 4: Backend: Endpoint GET /api/ciudad #1	10
Figura 5: Backend: Endpoint GET /api/ciudad #2	10

1 Características del Proyecto

1.1 Categoría

El proyecto se enmarca en la categoría de desarrollo de aplicaciones web, con un enfoque específico en el sector turístico.

1.2 Temática y Descripción

El tema central del proyecto es «**Turismo Ec**». La aplicación estará diseñada para proporcionar información detallada sobre las rutas turísticas en las ciudades, bosques y áreas protegidas de Ecuador. Además, presentará guías o rutas, los sitios más votados y los sitios no recomendados, proporcionando así una plataforma integral para los turistas que desean explorar Ecuador.

1.3 Alcance

El alcance del proyecto incluye el desarrollo de una aplicación web funcional que permita a los usuarios explorar diversas rutas turísticas en Ecuador. Ésta tendrá una interfaz amigable y estará diseñada pensando en el turista. La aplicación permitirá a los usuarios explorar diferentes rutas turísticas categorizadas como Ciudad, Bosque y Área Protegida.

Una de las características principales de nuestra aplicación será una funcionalidad de búsqueda bajo ciertos parámetros. Esto permitirá a los usuarios encontrar lugares específicos según sus preferencias, como el tipo de entorno (ciudad, bosque, área protegida), entre otras.

1.3.1 Historias de Usuario

1. «Como usuario, quiero poder buscar rutas turísticas en Ecuador filtradas por ciudades, bosques y áreas protegidas.»
2. «Como usuario, quiero poder dar mi opinión sobre la ruta seleccionada por comentarios y votación.»
3. «Como usuario, quiero poder añadir rutas como sitios turísticos, áreas protegidas y bosques.»
4. «Como usuario, quiero poder eliminar rutas y comentarios seleccionados.»

1.4 Languages de Programación y Paradigmas

Para el desarrollo de este proyecto, se utilizarán dos lenguajes de programación: **Ruby** para el backend y **JavaScript (Typescript)** para el frontend.

1.4.1 Backend

Ruby [1] es un lenguaje de programación de alto nivel que se centra en la simplicidad y la productividad. Para el backend de nuestra aplicación, utilizaremos **Sinatra** [2], una biblioteca de **Ruby** para la creación de aplicaciones.

Sinatra no sigue el patrón de diseño modelo-vista-controlador (MVC) que sigue **Ruby on Rails** [3]. En lugar de esto, **Sinatra** se centra en «crear rápidamente aplicaciones web en Ruby con un esfuerzo mínimo».

Sinatra ofrece una arquitectura *orientada a componentes* que proporciona más flexibilidad a los desarrolladores. Esto también indica que podemos tener una aplicación web moderna y completamente funcional con un código fuente mínimo y un solo archivo.

Además, para manejar la persistencia de datos en nuestra aplicación, utilizaremos **Firebase** [4], una base de datos no relacional proporcionada por **Google**. **Firebase** ofrece una variedad de servicios, incluyendo autenticación, almacenamiento y una base de datos en tiempo real.

1.4.2 Frontend

Para el frontend de nuestra aplicación, utilizaremos **JavaScript** [5], un lenguaje de programación de alto nivel que es especialmente eficaz para el desarrollo de aplicaciones web. **JavaScript** es conocido por su flexibilidad y es compatible con varios paradigmas de programación, incluyendo la *programación orientada a objetos* y la *programación funcional*.

Para mejorar la eficiencia y la seguridad de nuestro desarrollo frontend, utilizaremos **Deno** [6], un entorno de ejecución (*runtime*) para **JavaScript** con soporte nativo a **TypeScript** [7] que se centra en la

seguridad y la productividad. Junto a **Deno**, utilizaremos **Fresh** [8], un framework para **Deno** que nos permite crear aplicaciones web modernas con un esfuerzo mínimo.

El frontend de nuestra aplicación se comunicará con el backend mencionado anteriormente, esto a través de una *API REST*.

2 Prototipo

El prototipo de baja fidelidad para este proyecto se centrará en la definición de las características y funcionalidades clave de la aplicación, así como en la estructura general de la interfaz de usuario. Este prototipo se utilizará como una guía visual durante la fase de desarrollo de la aplicación.

Éste será diseñado en la plataforma **Figma** [9], a continuación la sección pertinente.

2.1 Prototipo de Baja Fidelidad en Figma

Se puede acceder a este prototipo a través del siguiente enlace:

«[Figma: Prototipo de Baja Fidelidad](#)»

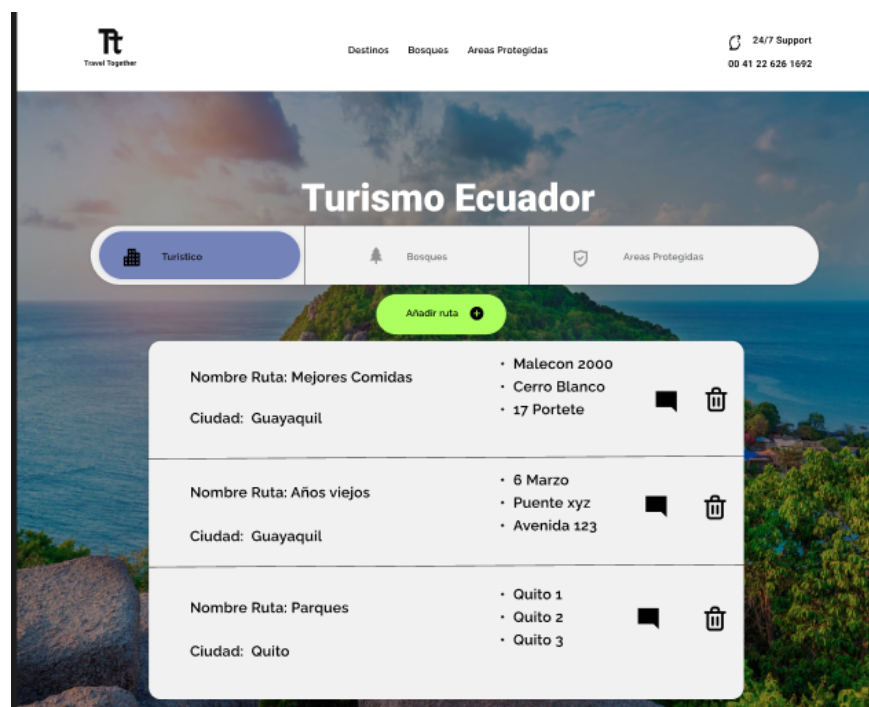


Figura 1: Prototipo de Baja Fidelidad - Pantalla 1

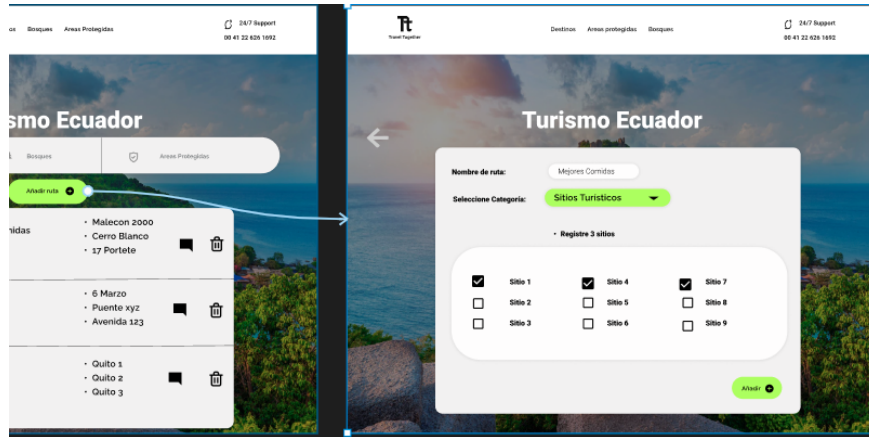


Figura 2: Prototipo de Baja Fidelidad - Pantalla 2

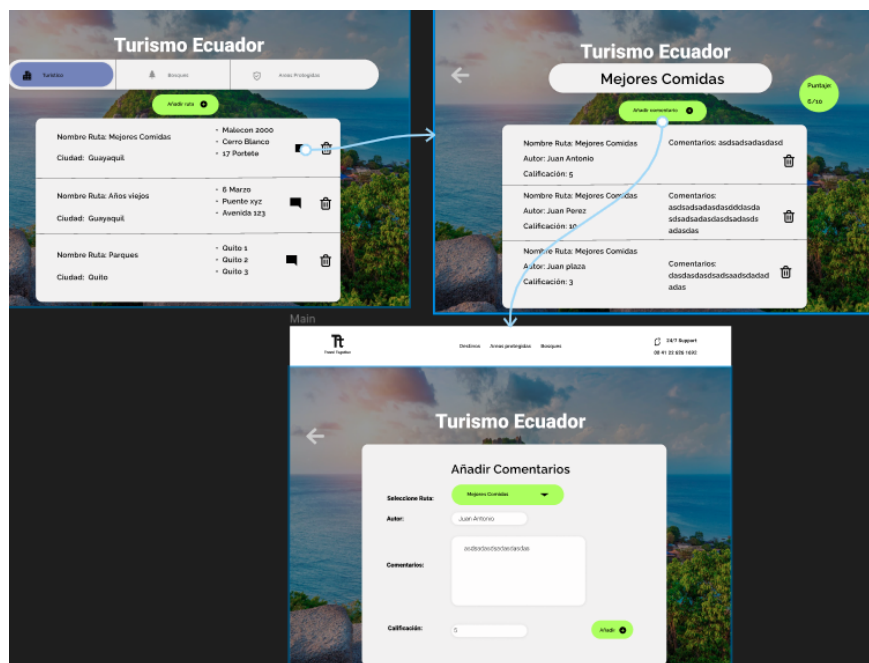


Figura 3: Prototipo de Baja Fidelidad - Pantalla 3

3 Gestión del Proyecto

Hemos decidido utilizar la funcionalidad «**DevContainers**» [10] de **GitHub** para gestionar el versionamiento de las tecnologías utilizadas en el desarrollo de nuestra aplicación.

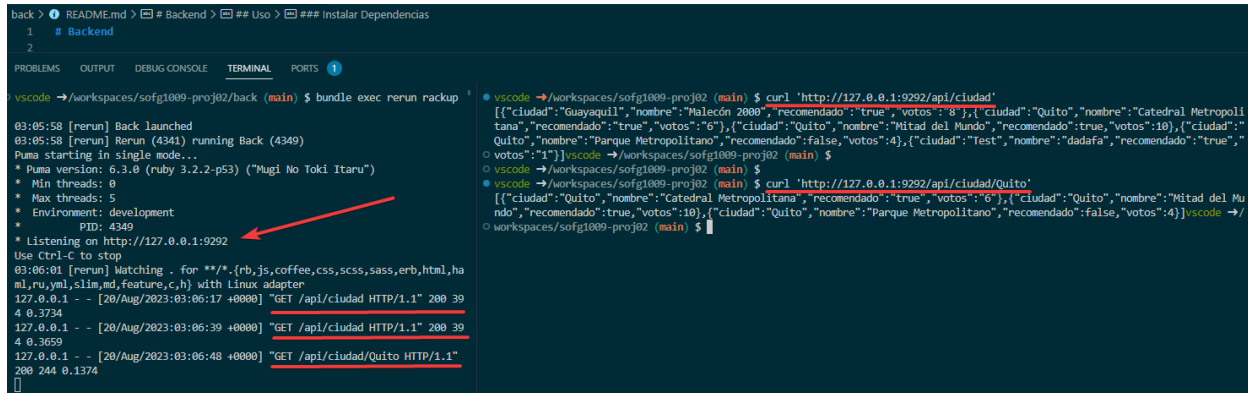
Un **DevContainer** es una configuración de entorno de desarrollo que se almacena en la nube. Permite personalizar un proyecto de **GitHub Codespaces** al incluir archivos de configuración en el repositorio, lo que crea una configuración de codespace reproducible para todos los usuarios del proyecto. Los **DevContainers** permiten automatizar una parte de tu entorno de desarrollo y facilitan el inicio del proceso de codificar

En resumen, la utilización de **DevContainers** nos permitirá gestionar de manera eficiente el versionamiento de las tecnologías utilizadas en nuestro proyecto, proporcionando un entorno de desarrollo coherente y estandarizado para todos los desarrolladores del equipo.

4 Desarrollo

4.1 Backend

4.1.1 Endpoint: GET /api/ciudad



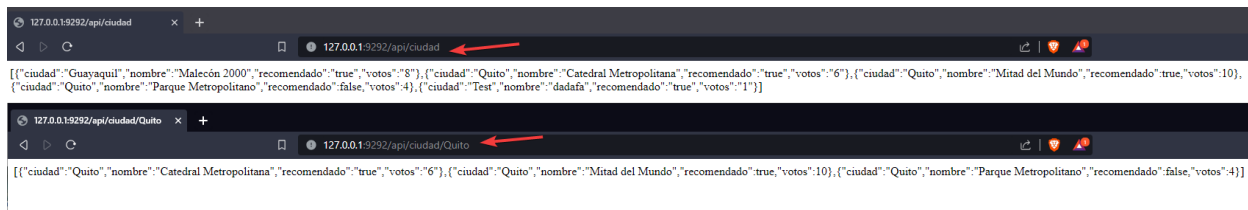
```
back > README.md > # Backend > ## Uso > ### Instalar Dependencias
1 # Backend
2

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ⓘ

vscode → /workspaces/sofg1009-proj02/back (main) $ bundle exec rerun rackup
03:05:58 [rerun] Back launched
03:05:58 [rerun] Rerun (4341) running Back (4349)
Puma starting in single mode...
* Puma version: 6.3.0 (ruby 3.2.2-p53) ("Mugi No Toki Itaru")
* Min threads: 0
* Max threads: 5
* Environment: development
* PID: 4349
* Listening on http://127.0.0.1:9292
Use Ctrl-C to stop
03:06:01 [rerun] Matching . for **/*.{rb,js,coffee,css,scss,sass,erb,html,haml,ru,yml,slim,md,feature,c,h} with Linux adapter
127.0.0.1 - - [20/Aug/2023:03:06:17 +0000] "GET /api/ciudad HTTP/1.1" 200 39
4 0.3734
127.0.0.1 - - [20/Aug/2023:03:06:39 +0000] "GET /api/ciudad HTTP/1.1" 200 39
4 0.3659
127.0.0.1 - - [20/Aug/2023:03:06:48 +0000] "GET /api/ciudad/Quito HTTP/1.1"
200 244 0.1374
[]

vscode → /workspaces/sofg1009-proj02 (main) $ curl 'http://127.0.0.1:9292/api/ciudad'
[{"ciudad":"Guayaquil","nombre":"Malecón 2000","recomendado":true,"votos":8}, {"ciudad":"Quito","nombre":"Catedral Metropolitana","recomendado":true,"votos":6}, {"ciudad":"Quito","nombre":"Mitad del Mundo","recomendado":true,"votos":10}, {"ciudad":"Quito","nombre":"Parque Metropolitano","recomendado":false,"votos":4}, {"ciudad":"Test","nombre":"dadafa","recomendado":true,"votos":1}]
vscode → /workspaces/sofg1009-proj02 (main) $
vscode → /workspaces/sofg1009-proj02 (main) $ curl 'http://127.0.0.1:9292/api/ciudad/Quito'
[{"ciudad":"Quito","nombre":"Catedral Metropolitana","recomendado":true,"votos":6}, {"ciudad":"Quito","nombre":"Mitad del Mundo","recomendado":true,"votos":10}, {"ciudad":"Quito","nombre":"Parque Metropolitano","recomendado":false,"votos":4}]
vscode → /workspaces/sofg1009-proj02 (main) $
```

Figura 4: Backend: Endpoint GET /api/ciudad #1



```
127.0.0.1:9292/api/ciudad x +
127.0.0.1:9292/api/ciudad
[{"ciudad":"Guayaquil","nombre":"Malecón 2000","recomendado":true,"votos":8}, {"ciudad":"Quito","nombre":"Catedral Metropolitana","recomendado":true,"votos":6}, {"ciudad":"Quito","nombre":"Mitad del Mundo","recomendado":true,"votos":10}, {"ciudad":"Quito","nombre":"Parque Metropolitano","recomendado":false,"votos":4}, {"ciudad":"Test","nombre":"dadafa","recomendado":true,"votos":1}]

127.0.0.1:9292/api/ciudad/Quito x +
127.0.0.1:9292/api/ciudad/Quito
[{"ciudad":"Quito","nombre":"Catedral Metropolitana","recomendado":true,"votos":6}, {"ciudad":"Quito","nombre":"Mitad del Mundo","recomendado":true,"votos":10}, {"ciudad":"Quito","nombre":"Parque Metropolitano","recomendado":false,"votos":4}]
```

Figura 5: Backend: Endpoint GET /api/ciudad #2

5 Anexos

5.1 Código Fuente

El código fuente de este proyecto se encuentra disponible en el siguiente [enlace a GitHub](https://github.com/anntnzb/sofg1009-proj02)

<<https://github.com/anntnzb/sofg1009-proj02>>.

Bibliografía

- [1] Ruby, “Ruby programming language.” <https://www.ruby-lang.org/>
- [2] Sinatra, “Sinatra: ruby web framework.” <http://sinatrarb.com/>
- [3] R. on Rails, “Ruby on rails.” <https://guides.rubyonrails.org/>
- [4] Google, “Firebase documentation.” <https://firebase.google.com/docs/>
- [5] M. D. Network, “Javascript.” <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
- [6] Deno, “Deno: a secure runtime for javascript and typescript.” <https://deno.land/>
- [7] Microsoft, “Typescript.” <https://www.typescriptlang.org/>
- [8] Fresh, “Fresh: the minimal, fast, and zero configuration deno web framework.” <https://fresh.deno.dev/>
- [9] Figma, “Figma: the collaborative interface design tool.” <https://www.figma.com/>
- [10] GitHub, “Github devcontainers.” <https://github.com/features/codespaces>