Aplicación web para calcular el índice de calidad de vida, en función del interés de un usuario con fines migratorios

Memoria de Proyecto Final de Máster

Máster Universitario en Desarrollo de sitios y aplicaciones web

Itinerario profesional

Autor: Manuel García López

Profesor: César Pablo Córcoles Briongos

Profesor: Julià Minguillón Alfonso

Profesor: Carlos Caballero González

8 de Mayo de 2020

Créditos

Licencias

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

Abstract

El objetivo de este proyecto es el desarrollo y documentación de una aplicación web a través de la que calcular el índice de calidad de vida de una manera más útil para el usuario, a través de ofrecer diversos parámetros de interés, que cada usuario puede decidir agregrar al cálculo de dicho índice o no. Esto permite que cada usuario pueda, según sus preferencias conocer cual es la calidad de vida de otras regiones del mundo y compararla con la suya.

Esta aplicación trata también de contar con un diseño responsive, para que se pueda adaptar a todos los tamaños de pantalla disponibles en ordenadores, móviles o tabletas.

En el desarrollo, se ha utilizado fundamentalmente Laravel como framework en Back-end y Angular como framework en Front-end.

Palabras clave: keywords del trabajo separadas por comas. Por ejemplo para éste documento podrían ser Modelo, Pauta, Memoria, Trabajo de Final de Grado/Máster, Word 2007

Palabras clave:
Aplicación web, Calidad de vida, Índice, Interés del usuario, Laravel, Angular

Abstract (english version)

The objective of this project is the development and documentation of a web application through which to calculate the quality of life index in a more useful way for the user, through offering various parameters of interest, which each user can decide to add when calculating said index or not. This allows each user to know, according to their preferences, to know the quality of life of other regions of the world and compare it with their own.

This application also tries to have a responsive design, so that it can be adapted to all screen sizes available on computers, mobiles or tablets.

In development, Laravel has been used primarily as a Back-end framework and Angular as a Front-end framework.

Keywords: job keywords separated by commas. For example for this document could be Model, Guideline, Memory, Final Degree Project / Master, Word 2007

Keywords:
Web application, Quality of life, Index, User interest, Laravel, Angular

Agradecimientos

Gracias a mi familia, especialmente a mis padres, porque sin ellos no hubiera podido tener la opción de llegar hasta aquí.

Gracias a mis amigos, por su ayuda y su apoyo permanente para hacerme mantener la ilusión y la motivación.

Gracias a todos mis profesores de máster, que me han ayudado muchísimo a aprender y a mejorar profesionalmente; y en especial gracias a Carlos, porque sin él este proyecto no hubiera sido posible.

Manuel García López

Índice

1. Introducción y contexto del trabajo ……10
2. Descripción ……11
3. Objetivos ……12
3.1 Objetivos principales ……12
3.2 Objetivos secundarios ……12
4. Metodología ……13
5. Plataforma y herramientas de desarrollo ……14
6. Planificación, presupuesto y análisis de mercado ……15
6.1 Planificación y proceso de desarrollo de la aplicación ……15
6.2 Presupuestado del coste de la aplicación ……19
6.3 Análisis de mercado ……22
7. Contenidos y arquitectura de la aplicación ……25
7.1 Contenidos ……25
7.2 Arquitectura de la aplicación ……26
8. Diseño e implementación ……28
8.1 Recopilación y clasificación de datos ……28
8.2 Prototipado de la aplicación ……32
8.3 Desarrollo e implementación de la aplicación ……44
9. Seguridad de la aplicación ……50
9.1 Identificación y tratamiento de los riesgos de seguridad ……50
9.2 Bugs de la aplicación ……50
10. Puesta en marcha y análisis de resultados ……51
11. Conclusiones ……53
12. Recomendaciones futuras ……54
12.1 Características a añadir ……54
12.2 Características a mejorar ……54
12.3 Mejoras en el desarrollo ……55
12.4 Adaptación a formatos móviles ……55
13. Requisitos de instalación y uso ……56
14. Instrucciones de instalación ……57
15. Manual de uso ……58
Anexo 1. Entregables del proyecto ……60
Anexo 2. Código fuente (extractos) ……61
Anexo 3. Librerías/Código externo utilizado ……64
Anexo 4. Capturas de pantalla ……65
Anexo 5. Bibliografía ……68
Anexo 6. Vita ……70

Figuras y tablas

Índice de Figuras

Figura 6.1: Diagrama de Gantt …….17
Figura 7.1: Arquitectura de la aplicación …….26
Figura 8.1: Sondeo de datos a obtener para la web ……...28
Figura 8.2: Rango de edad del participante ……...32
Figura 8.3: Género del participante ……....32
Figura 8.4: Nivel de estudios del participante ……... 33
Figura 8.5: Vida fuera de la ciudad actual …… 33
Figura 8.6: Vida fuera del país actual ……... 33
Figura 8.7: Posible uso de esta herramienta en base a las perspectivas de futuro ……..34
Figura 8.8: Árbol de navegación de la aplicación web ……38
Figura 8.9: Diagrama de flujo de la aplicación ……..38
Figura 8.10: Página de inicio de la aplicación ……..39
Figura 8.11: Menú condensado de la aplicación ……..39
Figura 8.12: Menú extendido de la aplicación ……..40
Figura 8.13: Comparador de la aplicación ……..40
Figura 8.14: Explorar una ciudad de la aplicación ……..41
Figura 8.15: Página inicial de la aplicación en escritorio ……..42
Figura 8.16: Menú de la aplicación en escritorio ……..42
Figura 8.17: Comparador de la aplicación en escritorio ……..43
Figura 8.18: Explorar una ciudad de la aplicación en escritorio ……..43

Índice de tablas

Tabla 5.1: Software utilizado para desarrollar la aplicación ……14
Tabla 5.2: Hardware utilizado para desarrollar la aplicación ……14
Tabla 5.3: Web apps utilizadas para desarrollar la aplicación ……14
Tabla 6.1: Planificación de trabajo ……...17
Tabla 6.2: Costes humanos ……20
Tabla 6.3: Costes software ……20
Tabla 6.4: Componentes hardware ……..21
Tabla 6.5: Costes hardware ……..21
Tabla 6.6: Costes adicionales ……..21
Tabla 6.7: Costes de producción ……21
Tabla 6.8: Precio de venta ……22
Tabla 8.1. Cálculo del índice económico ……31
Tabla 8.2: Descripción de usuario objetivo ……...35

1. Introducción y contexto del trabajo

La mejora de los medios de comunicación y transporte producida en las últimas décadas, así como las facilidades diplomáticas, y permeabilidad de las fronteras internacionales en numerosos lugares del mundo; han causado un importante incremento de los flujos migratorios de media y larga duración, por motivos principalmente laborales; esta tendencia seguirá en aumento y acelerándose durante este siglo según diversas predicciones.

Esta serie de migraciones las cuales se producen cada vez a mayor escala, las personas realizan muchas veces la elección del destino final de forma ligeramente arbitraria y con un conocimiento escaso de la región o ciudad objetivo.

Casi siempre el ideal perseguido detrás de este tipo de migraciones, es el de la mejora de la calidad de vida actual. Pero esto nos puede conducir a un gran interrogante, ¿Qué es la calidad de vida?

La calidad de vida es un ideal conformado por la agrupación de numerosos factores. Sin embargo, no todas las personas dan importancia a los mismos factores, ni en el mismo grado. Por lo tanto, podemos deducir que la calidad de vida, es un parámetro subjetivo dependiente de cada sujeto, de ahí la importancia de realizar este índice de este modo en vez de seleccionar otros índices ya preestablecidos como el HDI[1] y el GGI[2].

Es por esto que se ha decidido elaborar una aplicación web orientada a potenciales emigrantes que deseen mudarse a otra ciudad o región con el objetivo de mejorar su calidad de vida. Entendiendo por calidad de vida, el índice conformado por los parámetros de la web que ellos deseen utilizar.

A lo largo de esta memoria se irán planteando los distintos interrogantes, así como las soluciones planteadas en las distintas fases del desarrollo del proyecto. Además se expondrán los diversos objetivos, metodologías y herramientas utilizadas, junto al proceso de desarrollo de principio a fin.

1 HDI: Human Development Index (Índice de desarrollo humano)
2 GGI: Good Governance Index (Índice de buena gobernanza)

2. Descripción

Para entender mejor la motivación de realizar el proyecto, debemos comprender el problema que se plantea.

En numerosas ocasiones, y cada vez con más frecuencia, tenemos la opción por motivos, principalmente laborales, de cambiar nuestro lugar de residencia. Esta decisión se suele tomar en base a numerosos factores, pero principalmente debido a la expectativa de una mejora en nuestra calidad de vida. Es aquí cuando nos enfrentamos a una serie de dudas: ¿Qué es la calidad de vida?, ¿Cómo podemos medir la calidad de vida de varias regiones para compararlas entre si?, y lo que es más relevante para este proyecto: ¿Es la calidad de vida un valor objetivo y fijo para la sociedad?

La calidad de vida es un concepto que alude a diversos niveles o índices que, combinados, generan una sensación de mayor o menor bienestar. Estos índices abarcan numerosas cuestiones, referentes a economía, naturaleza, política o bienestar personal entre otros. En otras palabras, si bien algunos índices son generales y evaluables en términos numéricos como lo son los índices económicos, también existen otros índices no evaluables numéricamente. Por ejemplo, el bienestar personal, el cual es un índice totalmente subjetivo y relativo a cada individuo particular.

Entonces, ¿cómo podríamos calcular un índice de calidad de vida? Para realizar dicha finalidad de una forma lógica y objetiva, deberíamos de eliminar todos aquellos valores que no sean enumerables, es decir todos aquellos similares al tipo “bienestar personal, índice de belleza o índice de personalidad”. Si se decidiesen aplicar estos valores, se tendrían que racionalizar y convertir a valores numéricos, para posteriormente ser aplicados como filtro a un índice de calidad de vida ya calculado, o ser aplicados al propio índice. Sin embargo, este problema añade una complejidad excesiva al proyecto, por lo que no será abordado por la aplicación.

Por tanto, lo que la aplicación de este proyecto buscará, será el cálculo de un índice combinado de calidad de vida, en base a datos con valores, objetivos, fiables y de carácter numérico. Es aquí cuando se plantea la última gran duda: ¿Es la calidad de vida un valor objetivo y fijo?

La realidad es que si bien este índice se basa en valores objetivos, no existe una combinación perfecta que agrade a todo el conjunto de la sociedad. La mejor prueba de ello está en dos de los índices ampliamente usados para medir esta calidad de vida, como son el HDI y el GGI. Ambos tratan de reflejar el mismo concepto, pero se basan en condiciones y números diferentes.

Entonces, ¿cuáles son los mejores valores a incluir en un índice combinado de este tipo?

La realidad es que la combinación perfecta de condiciones no existe a nivel de sociedad, ya que como individuos tendemos a pensar diferentes y a mostrar gustos diferentes. A veces incluso, individuos con los mismos gustos, tienen un orden de preferencia e importancia totalmente diferente respecto a dichos gustos.

En base a esto, la solución propuesta por este proyecto para el cálculo del índice de calidad de vida, es realizar un cálculo variable, ligado al interés de cada usuario particular. Este índice también podrá ser capaz de establecer distintos órdenes de preferencia o relevancia, de forma que se ajuste lo máximo posible a los gustos de cada individuo, sin perder en exceso la objetividad.

Partiendo de esta idea, este proyecto consistirá en la realización de una plataforma web, la cual se encargue de:
Dados una serie de datos de entrada, enviar una petición a una API back-end realizada por nosotros, que le devuelva los datos requeridos siempre y cuando estos estén disponibles. Estos datos serán procesados por el cliente o front-end y serán mostrados sobre un mapa o tabla en la web.

El diseño estará orientado a múltiples dispositivos, de modo que dicha aplicación resulte sencilla e intuitiva para todos los soportes del mercado (móvil, tableta u ordenador).

El objetivo general es por lo tanto el de ofrecer una herramienta que ayude a la toma de decisión de carácter migratorio en base a un índice de calidad de vida personal de cada usuario. Esta decisión puede estar relacionada con diversos ámbitos, aunque el principal para el que está pensado este proyecto, es el de migrar por motivos de oferta laboral.

Para finalizar el nombre comercial elegido será el de: “Exodus”, o éxodo en inglés, en referencia a la temática migratoria que envuelve el desarrollo de esta aplicación web.

3. Objetivos

El objetivo general del proyecto es conseguir el desarrollo de una aplicación web, que sea capaz de ayudar de forma sencilla e intuitiva a cada usuario a tomar una decisión tan importante y estratégica como la de mudarse a otra región o ciudad del país, o del mundo; con el objetivo de mejorar su calidad de vida. Todo esto siempre teniendo en cuenta el mayor número de sus intereses personales posibles, así como su orden de preferencia en relación a estos.

3.1 Objetivos principales

Los objetivos clave del proyecto se estructuran en los siguientes:

- Analizar y desarrollar una solución en base al problema principal planteado: “Cálculo del índice de calidad de vida”.
- Desarrollar una aplicación que permita mostrar una solución calculada en cuando al índice de calidad de vida en base a los intereses seleccionados por medio de diversos filtros en nuestra web por cada usuario.

3.2 Objetivos secundarios

En cuanto a los objetivos secundarios se pueden identificar los siguientes:

- Diseñar y desarrollar una web multiplataforma, capaz de adaptarse a todos los dispositivos del mercado de forma cómoda.
- Añadir una gran variedad de filtros, de forma que el usuario sea capaz de obtener el índice de calidad de vida más acorde a sus requisitos personales.
- Añadir métodos de comparación entre distintas ciudades o regiones, así como otros tipos de información adicional considerada relevante.
- Crear una interfaz de un uso sencillo y de una necesidad de aprendizaje mínima.
- Adecuar la dimensión de la aplicación a la planificación temporal y el presupuesto proyectado.
- Profundizar en el conocimiento de las diferentes tecnologías de desarrollo web, así como en la comunicación entre ellas y entre los apartados front-end y back-end.

4. Metodología

La metodología para el desarrollo de este proyecto, será la siguiente:

Para empezar se tratará de obtener la información en base a los requisitos, datos y herramientas necesarias para poder desarrollar esta aplicación.

Una vez habiendo recopilado toda la información necesaria, se procederá a un estudio y análisis de los diversos datos obtenidos para obtener la mejor solución posible en cuanto a nuestro problema de estudio.

Una vez realizado esto, se procederá a estudiar y prototipar la aplicación y su flujo de trabajo. Se crearán modelos de alta calidad de las diferentes pantallas del proyecto, tanto en formato móvil como en formato de ordenador.

Una vez validadas las diferentes acciones a realizar gracias a los prototipos. Se comenzará el trabajo y desarrollo de la aplicación, tanto en el lado Back-end como en el lado Front-end de la aplicación. El lado Back-end, al tratarse de una API de consulta, se podrá desarrollar en paralelo con el prototipado de la aplicación, ya que este corresponde únicamente a las interacciones con el cliente o Front-end.

Al acabar el apartado Front-end, se procederá al testeo, corrección de fallos y documentación de los resultados obtenidos.

Por último, se pondrá en marcha una segunda fase de testeo con usuarios reales, para ver el funcionamiento de la primera versión de nuestro proyecto en el mundo real. A continuación, se realizará la presentación de todo el proyecto y de su memoria.

5. Plataforma y herramientas de desarrollo

Para el desarrollo del proyecto se utilizarán los siguientes recursos tecnológicos:

Software	
IDE Desarrollo	Visual Studio Code
Servidor Local	Xampp, Laravel Server y Node Server
Servidor web	Servidor de la universidad (UOC)
Maquetado de la documentación	Libre Office
Diseño gráfico	Gimp y OpenStreetMaps
Creación de wireframes	JustInMind
Navegadores de pruebas	Google Chrome y Mozilla Firefox for Developers
Tecnologías Front-end	HTML5, CSS3, SASS, JavaScript y TypeScript a través de Angular
Tecnologías Back-end	Php a través de Laravel y sql a través de PhpMyAdmin

Tabla 5.1: Software utilizado durante el desarrollo de la aplicación

Hardware	
Ordenador particular	Lenovo Z50, i7-4510U

Tabla 5.2: Hardware utilizado durante el desarrollo de la aplicación

Web apps	
Control de versiones	GitHub
Respaldo de datos y documentación	Google Drive

Tabla 5.3: Web apps utilizadas durante el desarrollo de la aplicación

6. Planificación, presupuesto y análisis de mercado

6.1 Planificación y proceso de desarrollo de la aplicación

Estimamos una jornada laboral de 6 horas al día y suponiendo cinco días laborables por semana. Se han ignorado los días festivos no pertenecientes a fines de semana.

Nombre	Duración en días laborables	Inicio	Final	Horas
Inicio del proyecto	10 días	19/02/2020	03/03/2020	60 horas
Definición del problema	2 días			12 horas
Diseño de una solución al problema	2 días			12 horas
Estudio de las fuentes de datos disponibles	2 días			12 horas
Redacción de la primera parte de la memoria	4 días			24 horas
Recogida de datos y prototipado de la aplicación	21 días	04/03/2020	01/04/2020	126 horas
Análisis de los datos de interés para los usuarios	1 día			6 horas
Selección de los datos de interés	2 días			12 horas
Procesado y almacenamiento de los datos	2 días			12 horas
Diseño del árbol de navegación	1 día			6 horas
Diseño de los diagramas de	4 días			24 horas

usabilidad UX				
Diseño del prototipo Hi-Fi	6 días			36 horas
Modelado de la base de datos	½ día			3 horas
Modelado de la arquitectura de la aplicación	½ día			3 horas
Redacción y revisión de la memoria	4 días			24 horas
Desarrollo de la aplicación	27 días	02/04/2020	10/05/2020	174 horas
Configuración del entorno de trabajo	1 día			6 horas
Programación de la aplicación	18 días			108 horas
Maquetación y retoques de estilos	3 días			18 horas
Revisión y correción de bugs y fallos de seguridad	½ día			3 horas
Creación del vídeo de testeo de la aplicación	½ día			3 horas
Redacción y revisión de la memoria	4 días			24 horas
Fase final del proyecto	21 días	11/05/2020	08/05/2020	126 horas
Puesta en marcha y testeo de la aplicación	12 días			72 horas
Análisis de resultados de los tests	1 día			6 horas

Redacción y revisión de la memoria	6 días			36 horas
Creación de la presentación del proyecto	2 días			12 horas
Total: 486 horas				

Tabla 6.1: Planificación de trabajo.

6.1.1 Diagrama de Gantt

En este caso, como se va a mostrar en el diagrama de Gantt, tenemos un flujo de trabajo secuencial, esto es debido a que solo vamos a contar con una persona trabajando en este proyecto para estos cálculos. Si el trabajo se realizase de forma grupal, se podrían paralelizar diversas tareas, con lo que se podrían reducir los tiempos de desarrollo.

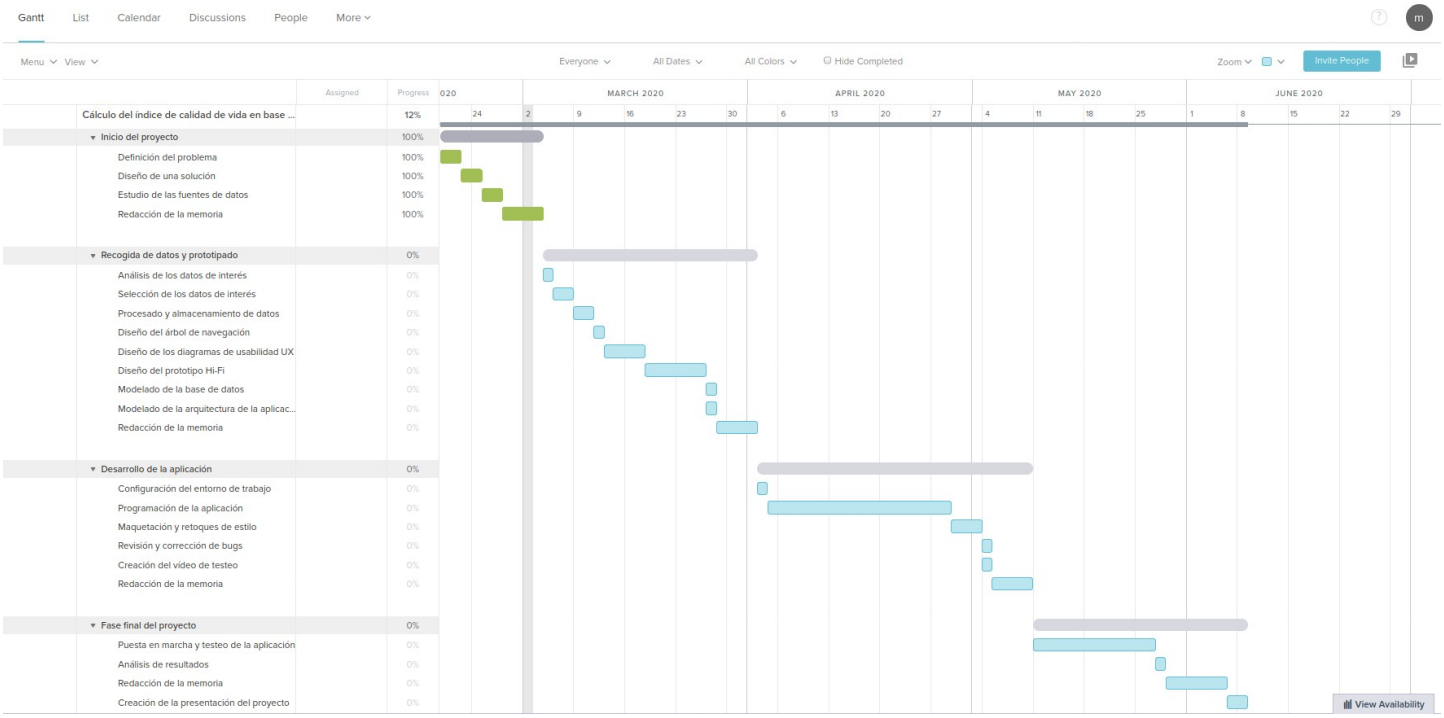


Figura 6.1: Diagrama de Gantt

6.1.2 Fases del desarrollo

Las fases a cumplir, con sus respectivas tareas serán las siguientes, y se podrán desarrollar en paralelo en caso de desarrollarse en grupo:

1- Definición del problema y planteamiento de la solución.

- Concretar cual es el problema al que nos enfrentamos
- Diseñar una solución lo más global y eficaz posible, sin perder de vista la funcionalidad y eficiencia de la misma.

2- Recopilación y almacenamiento de datos.

- Estudiar qué datos tenemos disponibles a distintos niveles (Nacional y municipal), y confirmar que las fuentes de los mismos son fiables.
- Seleccionar aquellos datos que sean útiles para el usuario general.
- Procesar y almacenar dichos datos, descartando todos aquellos que no nos sirvan de utilidad, a fin de reducir el tamaño de la muestra al mínimo necesario, siendo de este modo, más eficientes en términos de memoria.

3- Prototipado de una solución.

- Realizar una encuesta que nos ayude a determinar cuales pueden ser los datos más interesantes para el público general.
- Estudiar cual es el objetivo o perfil general de nuestro usuario.
- Enfocar un diseño y estructura de la web acorde al usuario planteado.
- Prototipar una web con JustInMind, acorde a la información recopilada anteriormente.

4- Desarrollo de una Base de datos y una API para el Back-end de la plataforma.

- Crear una base de datos de tipo sql a la que realizar peticiones desde nuestra API. (Phpmyadmin).
- Desarrollar una API en el lado del servidor con PHP, para que nos permita procesar y responder a cada cliente en base a las peticiones que este realice.

5- Desarrollo de una aplicación en el lado del cliente (Front-end) de la plataforma.

- Desarrollar una aplicación que se conecte con nuestra API (Back-end), recepcione los datos pedidos por el usuario (en base a diferentes interacciones) y los procese para mostrarlos de la forma más adecuada (En tablas o sobre un mapamundi).

6- Puesta en marcha de la plataforma web, y fase de pruebas.

7- Desarrollo de una memoria documentada del proyecto, en donde se deje constancia del desarrollo de las fases anteriores.

8- Entrega de la documentación, archivos, datos y código del proyecto.

6.2 Presupuestado del coste de la aplicación

Para presupuestar el coste total que esta aplicación tendría en la realidad, es necesario tener en cuenta los costes humanos, costes software, costes hardware y costes de recursos adicionales.

-Costes humanos

Para calcular los costes humanos asociados al proyecto vamos a suponer que, como en este caso, dispondremos de un único programador junior que elaborará todo el proyecto. Por lo que vamos a estimar el coste de su contratación de cara a la duración total del proyecto.

Los datos de sueldos han sido recogidos del sueldo medio en Madrid gracias al portal de búsqueda de empleo indeed, y podrían variar según la ciudad. Estos datos se han dividido en las catorce pagas habituales.

Es importante tener en cuenta que no todo el año es laborable, por lo que se tendrán en cuenta los siguientes días en los que no se podrá trabajar:
1- 30 días de vacaciones por año
2- 48 sábados al año (excluyendo los incluidos en las vacaciones)
3- 48 domingos al año (excluyendo los incluidos en las vacaciones)
4- 14 días festivos por año
Sumando todos estos días se obtiene un total de 140 días no laborables, los cuales restados al total de días anuales dentro de un año natural (365), nos dará 225 días hábiles al año por trabajador.

Programador Web Junior (1714,29€ mensuales)

Gastos ssgg
1- Contingencias comunes: 23,6% (de la nómina mensual) = 404,57€
2- Prestaciones por desempleo: 5,5% (de la nómina mensual) = 94,29€
3- Accidentes laborales o enfermedad profesional: 3,5% (de la nómina mensual) = 60€
4- Formación profesional: 0,6% (de la nómina mensual) = 10,29€
5- FOGASA (Quiebra de la empresa): 0,9% (de la nómina mensual) = 15,43€
Total ssgg: 584,58€

Total por trabajador mensual: 2298,87€

Puesto profesional	Sueldo	Tiempo de trabajo	Total
Programador web junior	32184,18€ anuales 23,84€ por hora	486 horas	11586,24€
Total		**11586,24€**	

Tabla 6.2: Costes humanos

-Costes software

En este apartado se va a tener en cuenta cual será el coste del software a utilizar en el desarrollo del proyecto:

Software	Unidades	Precio unitario	Tiempo de uso	Total
Sistema Operativo Ubuntu 18.04	1	0€	486 horas	0€
Visual Studio Code	1	0€	486 horas	0€
PHP 7.3	1	0€	486 horas	0€
Libre Office 6.0	1	0€	486 horas	0€
Mozilla Firefox Developer Edition 74	1	0€	486 horas	0€
Google Chrome	1	0€	486 horas	0€
GitHub	1	0€	486 horas	0€
Node.js	1	0€	486 horas	0€
Angular	1	0€	486 horas	0€
HTML, CSS, JS	1	0€	486 horas	0€
Total			**0€**	

Tabla 6.3: Costes software

-Costes hardware

En los costes hardware, incluiremos el presupuesto y las características de las máquinas a utilizar durante el desarrollo de la aplicación.

Componentes	Especificaciones
Procesador	Intel core i7-4510U 2.00GHZ
Memoria RAM	16 GB
Gráfica	Nvidia GEFORCE 860 M
Disco duro	1TB HDD

Tabla 6.4: Componentes hardware

Software	Unidades	Precio unitario	Tiempo de uso	Total
Ordenador personal	1	850€	486 horas	850€
Total			**850€**	

Tabla 6.5: Costes hardware

-Costes de recursos adicionales

En cuanto a los costes finales, tendremos en cuenta los diversos servicios utilizados, así como los recursos consumidos en material de oficina o fungible.

Servicio	Coste mensual	Tiempo de uso	Total
Internet de fibra óptica	50€	4 meses	200€
Servicio de luz	25€	4 meses	100€
Material fungible	0€	4 meses	0€
Total		**300€**	

Tabla 6.6: Costes adicionales

-Costes totales y precio de venta

Así mismo, tras calcular los costes parciales, podremos pasar a calcular los costes totales de producción del proyecto, así como el posible presupuesto de venta, para el cual estimaremos un beneficio del 6%.

Categoría	Coste
Costes humanos	11586,24€
Costes software	0€

Costes hardware	850€
Costes adicionales	300€
Total	**12736,24€**

Tabla 6.7: Costes de producción

Categoría	Coste
Coste inicial	12736,24€
Beneficio 6%	764,17€
Total sin IVA	13500,41€
IVA 21%	2835,09€
Total con IVA	**16335,50€**

Tabla 6.8: Precio de venta

6.3 Análisis de mercado

6.3.1 Contexto social y de mercado

Es importante destacar que este proyecto surge debido a una necesidad existente durante el manejo de otras plataformas similares ya existentes en Internet. Estas plataformas serán por tanto la inspiración en el desarrollo del mismo, pero también la competencia cuando el proyecto esté finalizado en su primera versión, y listo para pasar a producción.

Las ideas fundamentales a través de la cual se han encontrado estas plataformas es la siguiente: ¿Cómo es la vida en otras ciudades del mundo? ¿Cual es su calidad de vida, sistema sanitario o niveles de contaminación?

6.3.2 Productos similares

Para responder a estar preguntas habría por lo general que buscar numerosa información para cada una de estas ciudades y sacar en base a dicha información nuestras propias conclusiones. Es aquí cuando en medio de dicha búsqueda aparecen las plataformas: Numbeo, Nomad List y Expatistan. Las cuales procedo a analizar:

Numbeo

Numbeo[3] es una plataforma web en la que puedes interactuar con diferentes mediciones de carácter objetivo, y ver los datos de dichas mediciones en numerosas ciudades representadas a lo largo de un mapamundi. También proporciona los datos de una ciudad en concreto, y una comparación entre dos ciudades.

La web es de uso gratuito y dispone de una API de consulta de pago que según el volumen contratado cuesta entre 200 y 1800 euros al mes.

Ventaja: Visión global del mundo

Problema: Los valores de los índices de las ciudades están basados en los datos de una ciudad en concreto. Es decir, la puntuación de cada ciudad, es una puntuación comparativa con la ciudad modelo de cada índice, la cual no es la misma para todos los índices. Esto en mi opinión genera cierta confusión además de dificultar la comparación de dos ciudades que no son la ciudad modelo.

Nomad List

Nomad list[4] es una web que te ofrece una serie de datos de numerosas ciudades del mundo. Además posee un buscador mediante filtros que el usuario puede establecer para encontrar su destino idóneo. También ofrece otros servicios, como poner en contacto a los diversos usuarios de su red interna.

La web no dispone de API, y es gratuita.

Ventaja: La personalización de búsqueda de las distintas ciudades y la oferta de datos de cada una de ellas.

Problema: La falta de visión global y la extrema subjetividad de ciertos filtros como: Happy, fun, easy to make friends o beautiful women.

Expatistan

Expatistan[5] es una web similar a Numbeo pero orientada fundamentalmente a la comunidad expat. Ofrece un comparador del costo de vida, un mapamundi por colores y un calculador de salario.

La web no dispone de API, y el calculo detallado del costo de vida y salario cuesta 45 euros.

3 Numbeo
4 Nomad List
5 Expatistan

Ventaja: La información se muestra más amigablemente que en Numbeo.

Desventaja: Las mismas que posee Numbeo, ya que los índices están calculados en base a Praga (República Checa). Además su base de datos incluye bastante menos datos de interés.

En base a estas plataformas, y más allá de los problemas de cada una de ellas; la pregunta básica es: Si lo que quiero es saber cómo se vive en una ciudad en concreto, sin importarme su relación con las demás, pero teniendo una visión global del mundo. ¿Cómo lo hago?

Es aquí donde entra en juego este proyecto.

6.3.2 Estudio de mercado

Nuestro mercado de usuarios es el que engloba a toda aquella persona que tenga opciones por motivos familiares, laborales o de viaje, de cambiar de residencia de forma temporal; y quiera tomar dicha decisión, en base a sus propios gustos o intereses, los cuales se verán reflejados en los diferentes índices de nuestra aplicación.

La diferencia con los competidores anteriormente nombrados, radicará en la procedencia del valor mostrado para cada ciudad. El cual no será en función a una ciudad modelo, sino a los propios valores de la propia ciudad. La razón de calcular los valores de este modo, es que es más sencillo representarlos en una escala del 0 al 100, más fácil de comprender para los usuarios; además, este método proporciona una nota local, la cual es mucho más interesante a muchos niveles, como por ejemplo el económico.

El objetivo final será el de mostrar todos estos valores en un mapamundi, para que a pesar de representar valores locales, se puedan comparar con valores de todo el mundo; para así poder determinar cual es la calidad de vida en cada una de esas ciudades.

7. Contenidos y arquitectura de la aplicación

7.1 Contenidos

La aplicación web estará estructurada de forma clásica, es decir, con una cabecera, un cuerpo y un pie de página.

a) Cabecera: En ella se encontrará el logo de la aplicación web, un enlace al comparador de ciudades y una serie de enlaces a cada uno de los distintos índices de datos que se encuentran disponibles por ciudad en la web.

b) Cuerpo: En esta parte se verán representados los diversos datos de cada una de las secciones de la web. Cada sección contendrá diversos y diferentes elementos.

c) Pie de página: Dispondrá de un enlace a una sección de información sobre el proyecto, e información sobre el copyright y autoría de la aplicación.

Esta estructura será la que se mantenga a lo largo de todo el proyecto, y en sus diversas secciones. Dichas secciones serán las siguientes:

a) Inicio: En dicha sección se mostrará por defecto un índice de calidad de vida calculado para las ciudades de las que se dispongan determinados datos. Además se mostrará una imagen con las ciudades con mejores resultados en dicho índice.

b) Listado de índices: En cada uno de ellos se mostrará información del mismo modo que para el inicio, con la salvedad de que se utilizarán datos diferentes en función del índice requerido por el usuario.

c) Comparador de ciudades: En esta sección se podrán comparar los valores de cada índice de dos ciudades diferentes entre sí, con el fin de ver en detalle las diferencias entre dicho valores. En este comparador se podrán comparar todas las ciudades disponibles en nuestra base de datos.

d) Listado de datos por ciudad: En este apartado se mostrará a información relativa a la ciudad seleccionada, y al país al que pertenece. Además mostrará su índice de calidad de vida, así como ofrecerá la posibilidad al usuario de recalcular dicho índice para dicha ciudad, solo en función de los valores que desee, y de la importancia que les dé a estos.

e) Sobre nosotros: Sección en la que se informará al usuario sobre diferentes aspectos de la web.

7.2 Arquitectura de la aplicación

La aplicación se compondrá de los elementos típicos que se pueden observar en la mayoría de plataformas web. Esto es, un Back-end (en el Servidor), un Front-end (navegador de Internet) y una base de datos.

Nuestro cliente o Front-end, será el encargado de realizar las diversas llamadas al Back-end, el cual estará implementado con un protocolo de comunicación RESTful[6]. Es decir, la información se intercambiará mediente el manejo de la API que habrá en el servidor. Esta API será la encargada de interpretar las peticiones del cliente mediante http, consultar la base de datos de la forma más adecuada, y devolver una respuesta al cliente. Por último, el cliente procesará los datos, y los mostrará al usuario.

Este tipo de comunicaciones, que corresponden al patrón MVC[7], se pueden representar con el siguiente esquema:

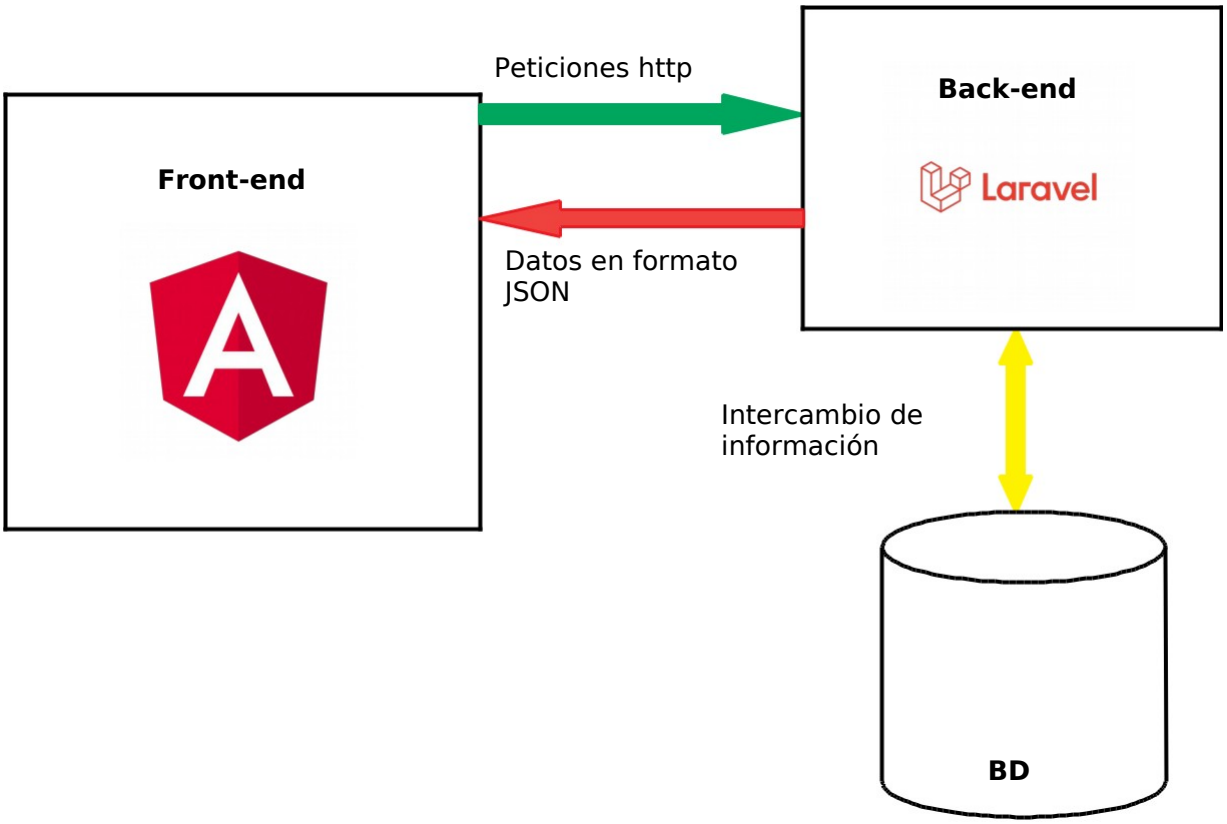


Diagrama 7.1 Arquitectura de la aplicación

En concreto, en este proyecto, cada una de las partes realizará las siguientes tareas:

6 RESTful: Tipo de conexión encargada de manejar datos a través de operaciones CRUD (Get, Post, Update y Delete)

7 MVC: Siglas que pertenecen al patrón Modelo, Vista, Controlador

a) Base de datos: Se desarrollará con PhpMyAdmin en lenguaje SQL, y se dedicará a almacenar toda la información en lo referente a nuestro proyecto.

b) Back-end: Se desarrollará con Laravel en lenguaje PHP, y se dedicará a intercambiar información con la base de datos, procesar las peticiones de los clientes, y responder con los datos en formato json; sirviendo de API.

a)

c) Front-end: Se desarrollará mediante Angular y los lenguajes de HTML, CSS y Typescript (JavaScript para los navegadores). Su función principal es la de pedir y recibir datos, así como procesarlos y mostrarlos en la interfaz de usuario.

Las características de nuestra API serán las siguientes:

1- Se tratará de una API con protocolo sin estado donde las diversas peticiones HTTP se traten de forma independiente.

2- A pesar de que una API suele estar formada por las cuatro operaciones CRUD, en esta API se van a realizar solo consultas. Es por ello, que la operación fundamental de la API será GET. Además el acceso se realizará mediante URI[8].

3- Contará con un sistema de autenticación para evitar accesos indebidos a dicha API que saturen el sistema.

Como característica adicional, se mantendrá un almacenamiento secundario a modo de caché con Local Storage. En el que se almacenen los datos que se prevean vayan a sufrir un mayor volumen de peticiones.

8 URI: Uniformed Resource Identifier (Identificador de recursos uniforme)

8. Diseño e implementación

8.1 Recopilación y clasificación de datos

8.1.1 Selección y clasificación de datos

La selección de datos se ha establecido tras realizar en primer lugar una comprobación de los recursos usados por parte de los competidores, y que siguieran un cierto criterio en cuanto a relevancia y en cuanto a objetividad.

Para destacar el grado de relevancia, se hizo además un sondeo a través de una encuesta, en la que se pretendía estudiar cuales de estos índices eran más importantes para los usuarios. Además, gracias a dicha encuesta se pudo realizar también un análisis del usuario objetivo. En esta pregunta se registraron 56 respuestas de entre 61 participantes.

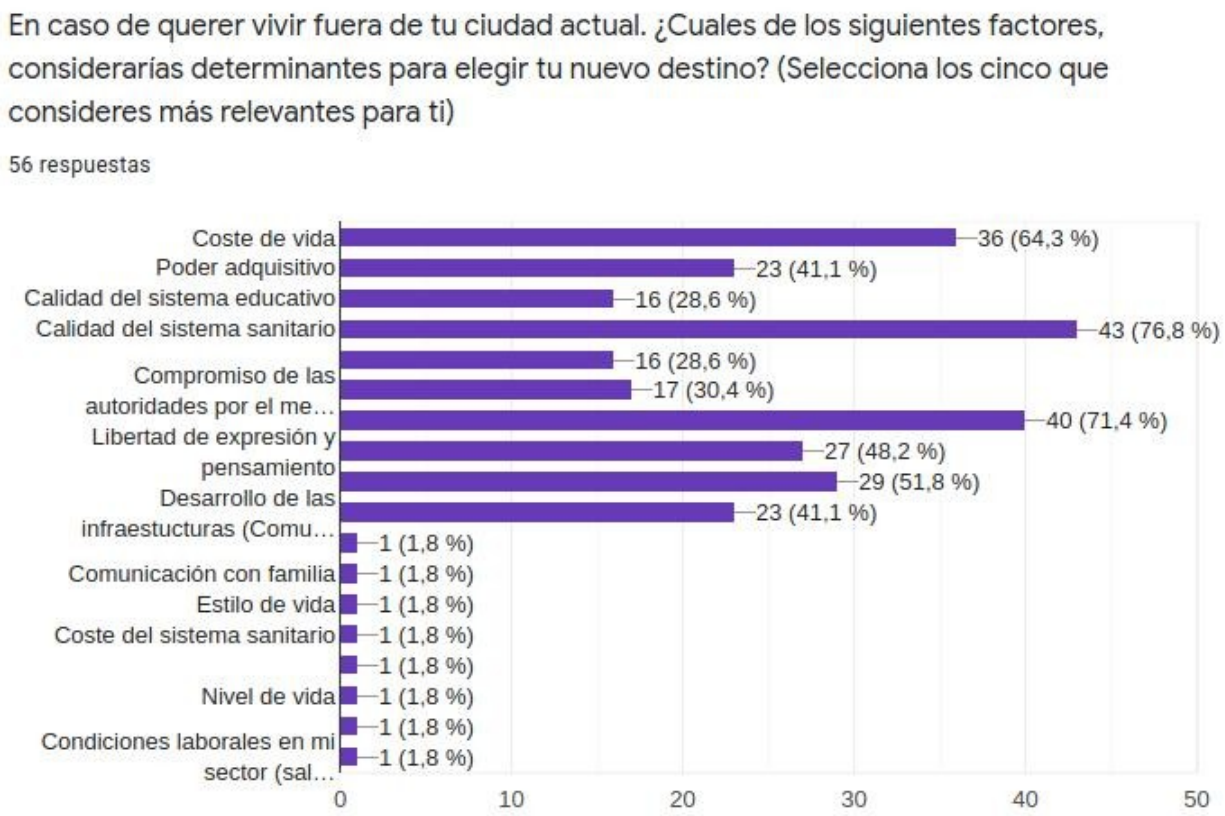


Figura 8.1: Sondeo de datos a obtener para la web

Como podemos observar, de entre los factores testeados, los más valorados por los usuarios son:

- Calidad del sistema sanitario

- Seguridad ciudadana

- Coste de vida

Y en menor medida, también resultan relevantes:

- Tasa de empleo
- Libertad de expresión y pensamiento
- Desarrollo de las infraestructuras
- Poder adquisitivo

Sin embargo, dado que la muestra no es determinante debido a su escaso tamaño, se han decidido incluir todos estos índices en el cálculo.

Esta decisión, se ha tomado también acorde al impacto social que hoy en día todos estos índices tienen en la sociedad, y a la posibilidad de que nuestros usuarios puedan descartar alguno de estos índices a la hora de usar calcular el índice de calidad de vida de una ciudad.

A continuación se procede a la búsqueda de dichos datos. Como ya se ha indicado anteriormente, Numbeo posee una API de pago, pero además ofrece la opción de uso gratuito para estudiantes. Es en base a esto, entorno a lo que se ha decidido descargar los datos que esa esa plataforma y que son de gran utilidad, a saber: Coste de vida, calidad del sistema sanitario, tasa de criminalidad, contaminación ambiental etc.

No obstante y pese a que estos datos podrían ser suficientes, no abarcan la totalidad de los factores evaluados en la encuesta. Para el resto de datos, se han usado los datos extraídos a nivel de país del portal mundial de datos abiertos “World bank open data”[9].

Desafortunadamente este portal no ha podido proporcionarnos datos suficientemente adecuados para establecer la medición de: “Libertad de expresión” y “Calidad educativa”. Es por esto y por la dificultad para encontrarlos por medio de otras fuentes, por lo que se ha decidido prescindir de ellos.

Es importante hacer hincapié en que los datos de Numbeo, se encuentran distribuidos en regiones o países y en ciudades. Mientras que los datos del World bank open data, se encuentran solo distribuidos por países.

Sin embargo esto no afectará a la calidad de nuestros datos, ya que los índices en los que se hará uso de los datos a nivel de país adquieren más sentido a dicho nivel que si fuera por ciudades.

9 World bank open data es un portal web que reune datos oficiales de todos los países del mundo, agrupados por numerosas temáticas.

8.1.2 Filtrado de los datos

Una vez seleccionados los diferentes datos, los filtraremos y fusionaremos has quedarnos con los siguientes índices:

1- Renta y poder adquisitivo

2- Desarrollo de las infraestructuras

3- Protección del medioambiente

4- Índice de criminalidad

5- Índice de empleo

6- Índice del tráfico

7- Protección a la mujer

8- Índice de diversidad cultural

9- Índice de contaminación

10- Calidad del sistema sanitario

11- Índice de calidad de vida

Adicionalmente dispondremos de la siguiente información por cada ciudad:

- Salario medio
- Nombre de la ciudad y país al que pertenece
- Coordenadas (latitud y longitud)

Nuestro dataset final, tendrá toda esa información almacenada en un único archivo con información sobre más de cinco mil ciudades de todo el mundo. Es importante especificar que inicialmente se disponía de catorce mil ciudades de todo el mundo, pero que debido a falta de información, esas ciudades se han reducido a cinco mil.

Además de estos datos, dispondremos de otro archivo en el que encontraremos los mismos campos de información que en el anterior archivo pero disponiendo de solo 424 ciudades. El criterio de selección de estas ciudades, es que de las anteriores cinco mil, estas son las únicas que tienen un salario medio propio, ya que el resto de ciudades poseen los valores de los salarios medios del país.

Por último encontraremos la información relativa a los países y ciudades (sin transformar a nuestra escala de índices) y la información relativa los valores de cambio de moneda (para poder expresar el valor del dinero según la moneda de cada país).

8.1.3 Procesado de los datos y cálculo de los diversos índices

Como ya se ha indicado, la nota que se va a mostrar por cada ciudad en los diversos índices, es la nota local, es decir, la nota que cada uno de los índices obtiene sobre los datos de su ciudad, y no sobre los datos de otra ciudad del mundo.

Para obtener estas notas a partir de los diversos datos disponibles se ha procedido a la normalización de estos en base a una escala del uno al cinco, incluyendo los valores enteros intermedios, donde uno es muy negativo y cinco muy positivo. Veamos un ejemplo:

Dadas dos ciudades, Madrid y Zúrich; un índice a analizar, el económico, y los valores de salario, cpi y renta, poder adquisitivo y tasas de ahorro en inflación nacionales; veamos que nota económica obtenemos según nuestro método.

Valores	Muy negativo (1)	Negativo (2)	Neutro (3)	Positivo (4)	Muy Positivo (5)
Salario	<500	<1000	<1500	<2000	>2000
Cpi y renta	>80	<80	<60	<40	<20
Poder adquisitivo	<25	<50	<75	<100	>100
Ahorro	<0	<10	<20	<30	>30
Inflación	>4	<4	<3	<2	<1

Tabla 8.1. Cálculo del índice económico

Suponiendo unos valores como los siguientes:

- Madrid (Salario: 1600, Cpi y renta: 55, Poder adquisitivo: 70, Ahorro: 12, Inflación: 0.5)
- Zúrich (Salario: 4400, Cpi y renta: 84, Poder adquisitivo: 104, Ahorro: 23, Inflación: 0.7)

Las notas serían las siguientes:

- Madrid: (4+3+3+3+5)/5 = 18/5 → 3,3*20 → 66
- Zúrich: (5+2+5+4+5)/5 = 21/5 → 4,1*20 → 82

Una vez aplicado este método a los distintos índices con sus diferentes datos, podemos obtener las notas de cada uno de ellos, y en función de cada uno de dichas notas, podemos construir el índice de calidad de vida de cada ciudad, simplemente sumando sus diferentes subíndices y dividiéndolo entre el número de subíndices.

8.2 Prototipado de la aplicación

8.2.1 Análisis del usuario objetivo

El usuario objetivo es aquel que según nuestros estudios, es aquel que debido a ciertas características, es el más propenso a hacer uso de esta aplicación. Para ello nos realizaremos una serie de preguntas de competencia, que tendrán que ser respondidas en una encuesta por una serie de participantes. En base a los resultados extraídos de esta encuesta, podremos realizar un análisis de los mismos, y determina a que tipo de usuario objetivo nos enfrentamos, para realizar un diseño de la web más acorde a dicho usuario.

-Estudio realizado

La encuesta realizada, fue respondida por 61 participantes. A continuación se muestran las preguntas de competencia realizadas y los resultados obtenidos:

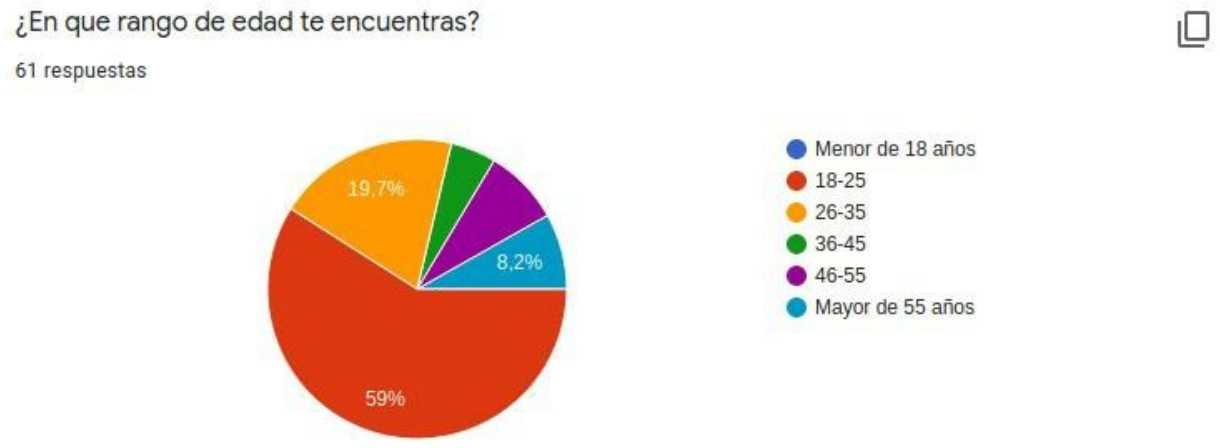


Figura 8.2: Rango de edad del participante

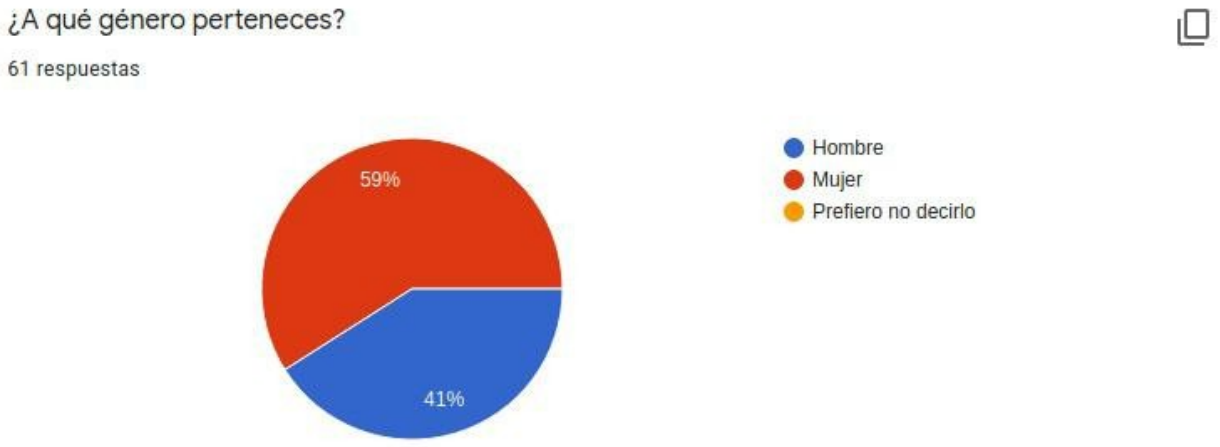


Figura 8.3: Género del participante

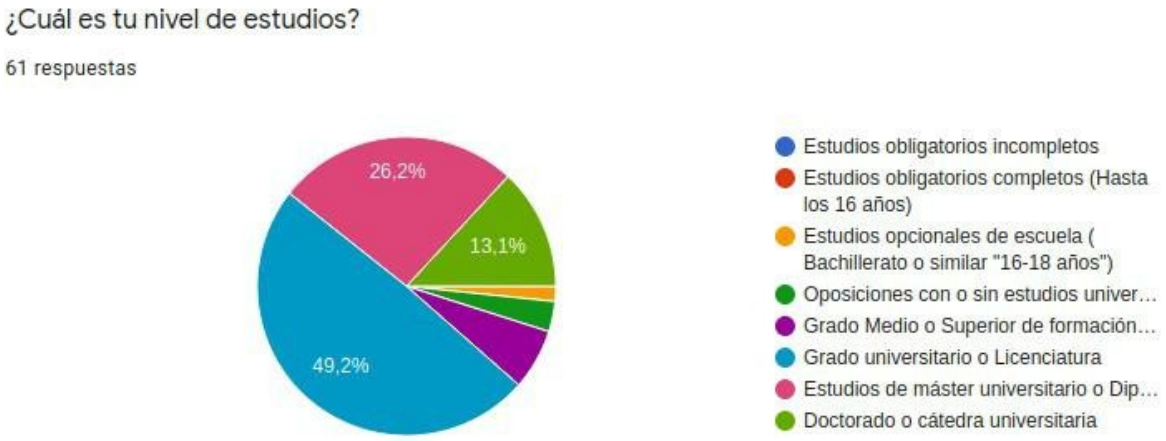


Figura 8.4: Nivel de estudios del participante

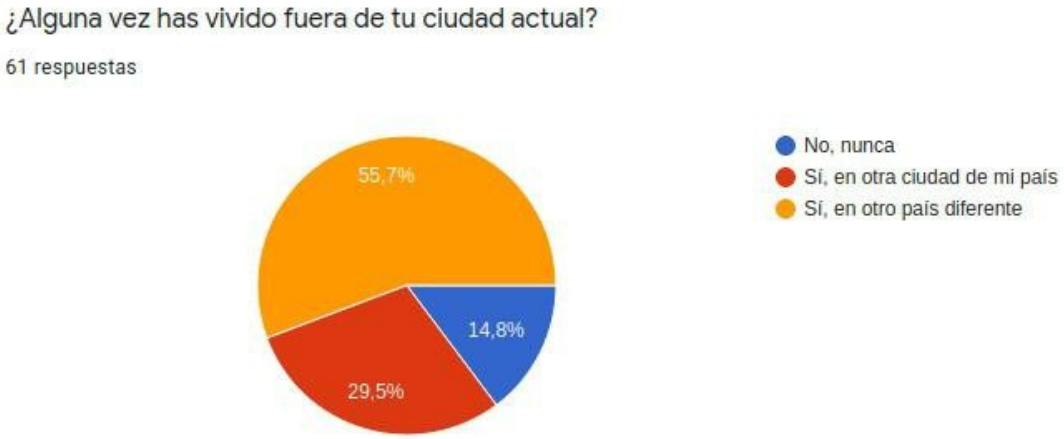


Figura 8.5: Vida fuera de la ciudad actual

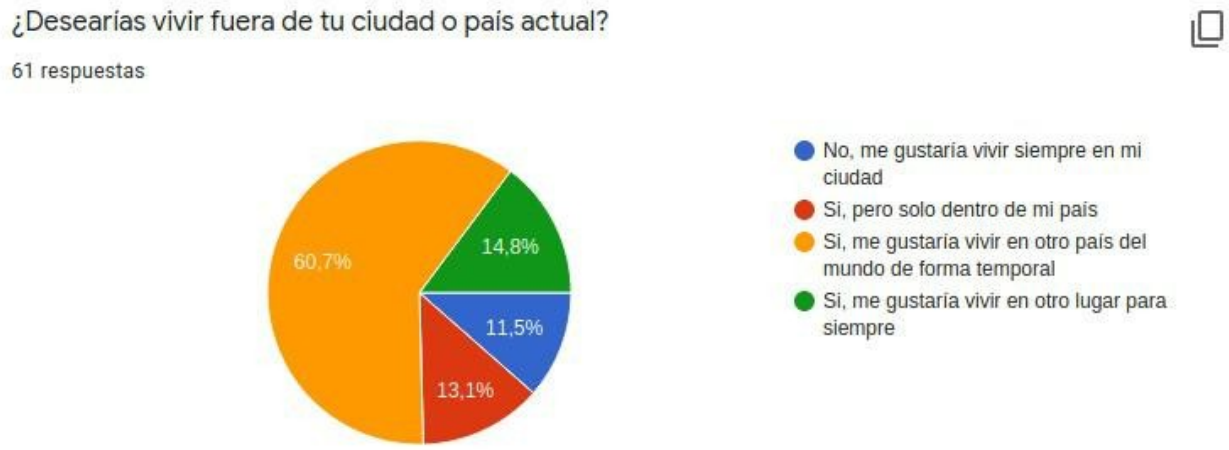


Figura 8.6: Vida fuera del país actual

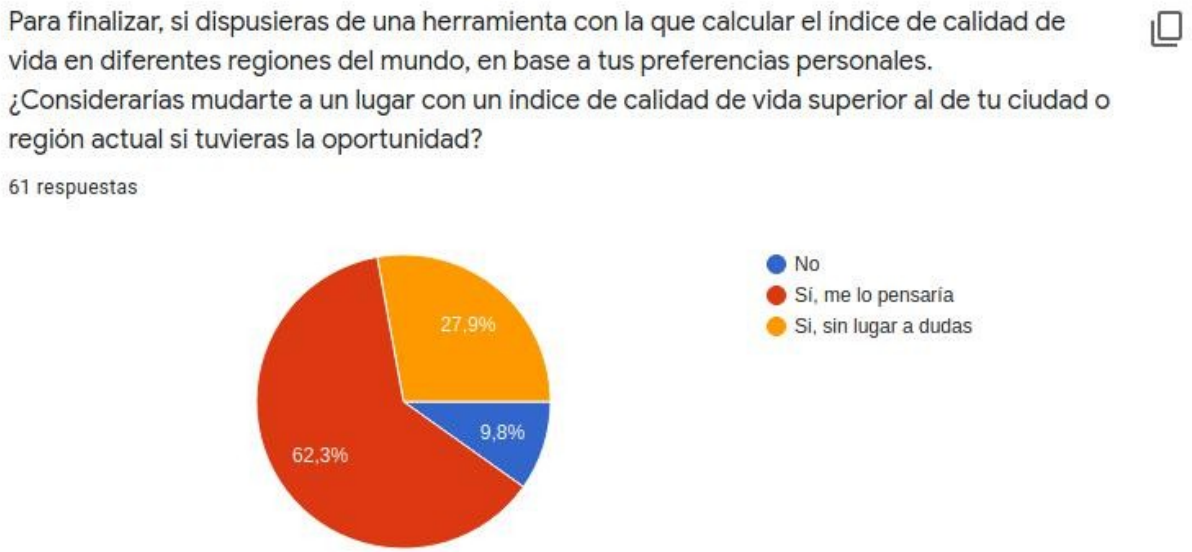


Figura 8.7: Posible uso de esta herramienta en base a las perspectivas de futuro

-Conclusiones del estudio

Según los datos obtenidos de este estudio, podemos concluir que nuestro usuario objetivo se trata de una persona joven, con estudios universitarios de cualquier rango, y que ya ha vivido fuera de su ciudad natal previamente.

-Defición de las personas objetivo y sus necesidades

Un ejemplo de potencial usuario de nuestra aplicación es el siguiente:

Nombre: Lucía
Localidad: Alicante, España
Rango de edad: 25-30 años
Profesión: Diseñador publicitario en Alicante.
Objetivos: Ser una gran profesional a nivel internacional.
Puntos débiles: Las situaciones nuevas le causan mayor inseguridad.
Cita personal: “El mundo no sería el mismo sin la publicidad”
Biografía: Llamar la atención de su entorno familiar ha sido desde niña su principal interés. Ahora, su interés se centra en realizar campañas para captar la atención de millones de espectadores.
Nivel de estudios: Máster en márketing y publicidad internacional
Salario actual: 25000 euros anuales
Aspiraciones de vida: Dar prestigio a la empresa para la que trabaje a través de sus campañas.
Hobbies: Pintura y fotografía escénica.

Preocupaciones laborales: No ser reconocida en su ámbito laboral.
Uso del móvil: Alto, le encanta navegar por Internet en busca de ideas e inspiración.
Uso de los Pc: Alto, principalmente en temas de edición de vídeo y fotografía.
Confort social: Socialmente cómoda entre grandes masas de personas.
Motivación: Innovar y mejorar la calidad de la publicidad.

Tabla 8.2: Descripción de usuario objetivo

-Contexto de acceso

El contexto de acceso del usuario objetivo, es el de un mundo cada vez más global, donde las sociedades de cada país o región, ya no trazan fronteras entre sí. Un mundo de libre movimiento de personas, en el que sin embargo, existen una gran multitud de inquietudes y miedos. Pero también un mundo lleno de ofertas para todos aquellos que estén dispuestos a aceptar el reto que estas suponen.

Persona tipo Lucía

Lucía ambiciona descubrir el mundo realizando campañas publicitarias en distintos lugares del mismo. Sin embargo, cambiar de lugar en el que vivir, aún contando con una oferta laboral, es un gran desafío lleno de interrogantes. Esta serie de interrogantes son la razón por la que la aplicación de este proyecto, representará una gran oportunidad para aclarar sus dudas y hacer que tome una importante decisión ante cada una de las diferentes oportunidades laborales.

8.2.2 Patrones de diseño y usabilidad

Antes de pasar al prototipado de la aplicación vamos a detallar los diversos patrones de usabilidad a utilizar en dicha aplicación:

-Patrones de diseño para obtención de datos

1- Valores por defecto (Good defaults)

Los formularios de la aplicación contendrán valores por decfecto correctos a modo de guía, con el fin de indicar de forma clara al usuario qué debe realizar en cada parte del formulario.

2- Explicativo (Wizard)

Todas las acciones, tanto de selección como formularios poseerán una aclaración sobre la acción que se va a realizar. Es decir, toda acción estará debidamente especificada.

-Patrones de navegación

1- Etiquetas de navegación (Navigation Tabs)

Este patrón se va a usar para la ordenación de los diferentes elementos de la web disponibles en los menús de navegación. Cada etiqueta contendrá un nombre corto que especificará información la sección de la aplicación a la que se da acceso prioritario.

2- Enlace al home (Home link)

Este patrón proporcionará un acceso al home o página de inicio desde el logotipo de la web.

3- Menú de despliege vertical (Vertical Dropdown Menu)

Este patrón se usará en el menú principal de la aplicación. Con este patrón, el menú se desplegará en un bloque vertical, que en caso de tratarse del escritorio del ordenador, ocupará todo el ancho de la pantalla. Mientras que en caso de móviles, se desplegará ocupando parcialmente uno de los laterales de la pantalla.

4- Categorización (Categorization)

Este patrón agrupará los diferentes elementos según las categorías principales de la web, en este caso, los diferentes índices y el comparador de ciudades. Es en base a estas categorías principales en torno a las que se establecerá el menú.

5- Scroll infinito (Continuous scrolling)

Este patrón garantiza que se pueda hacer scroll de forma infinita en lugar de una paginación de contenidos. Esto es así debido a que nuestra aplicación no contendrá un gran volumen de recursos que cargar, por lo tanto no será necesario establecer paginación.

-Patrones de procesado de datos

1- Inicio (Dashboard)

El uso de este patrón está orientado a dar una visión global de los datos y funcionalidades de la aplicación.

Con el fin de hacer entender al usuario mediante una primera impresión, el uso de la web.

2- Autocompletar (Autocomplete)

Este patrón ayudará al usuario a visualizar la información disponible en la base de datos conforme escriba en los campos del formulario. En nuestro caso al tratar de localizar una ciudad concreta.

-Patrones de abordaje

1- Pizarra en blanco (Blank slate)

Este patrón se usará para ayudar y motivar al usuario a realizar diversas acciones en la web, como comparar o buscar información de una ciudad. Para ello se hará uso de colores y dibujos llamativos.

-Patrones de cognición

1- Ilusión de control (Illusion of control)

Mediante este patrón, el usuario adquire conciencia de que la aplicación realiza acciones porque el usuario quiere que estas acciones sean realizadas. De este modo el usuario se encuentra más cómodo en el manejo de la aplicación. Esta ilusión se puede ver en el uso de autocompletar para los formularios, o en el cálculo del índice combinado de calidad de vida para una ciudad en concreta.

-Patrones de percepción y memoria

1- Reducción (Reduction)

Este patrón se basa en la simplificación de las tareas fundamentales que ha de hacer el usuario, de tal forma que este se enfoque en dichas tareas para entenderlas mejor. Se usará en los formularios de la web.

2- Reconocimiento de patrones (Pattern recognition)

Mediante este patrón hacemos que el usuario se familiarize rápidamente con el manejo de la web. Este patrón consiste en tener un número pequeño de interfaces diferentes, y hacer que estas sean lo más repetitivas posibles para activar la memoria a corto plazo del usuario y no obligarle a tener que entender cada una de las interfaces de la web. Se usará en los índices de la web.

-Patrones de diseño de la estructura

1- Layout

En el layout, haremos uso del patrón Mostly Fluid[10]. El cual se usará para ordenar la distribución de las diferentes columnas de contenido en nuestra aplicación. Esto es muy importante para crear una web con diseño responsive. Ya que si bien en escritorio, las columnas de contenido se estructurarán principalmente en horizontal, en formatos móviles o en tabletas, estas columnas se colapsarán para quedar unas debajo de otras en formato vertical.

2- Off Canvas[11]

El menú del móvil seguirá un patrón Off Canvas en el margen lateral izquierdo. Esto quiere decir que en formato móvil, el menú aparecerá y desaparecerá de la pantalla según lo requiera el usuario.

---

10 Mostly fluid: Patrón de diseño responsivo por el cual los contenidos se distribuyen por columnas en función del espacio disponible.

11 Off Canvas: Patrón de diseño móvil que esconde el contenido fuera de los límites de la pantalla, y lo muestra cuando el usuario lo requiere.

8.2.3 Árbol de navegación y diagrama de flujo

-Árbol de navegación

Para representar las diferentes pestañas de navegación, dispondremos del siguiente esquema, dividido en el dashboard o pantalla de inicio, el comparador de ciudades, y los subíndices que podemos explorar.

Figura 8.8: Árbol de navegación de la aplicación web.

-Diagrama de flujo

Por otro lado, el diagrama de flujo de la aplicación, nos permite visualizar los diferentes estados disponibles en la aplicación.

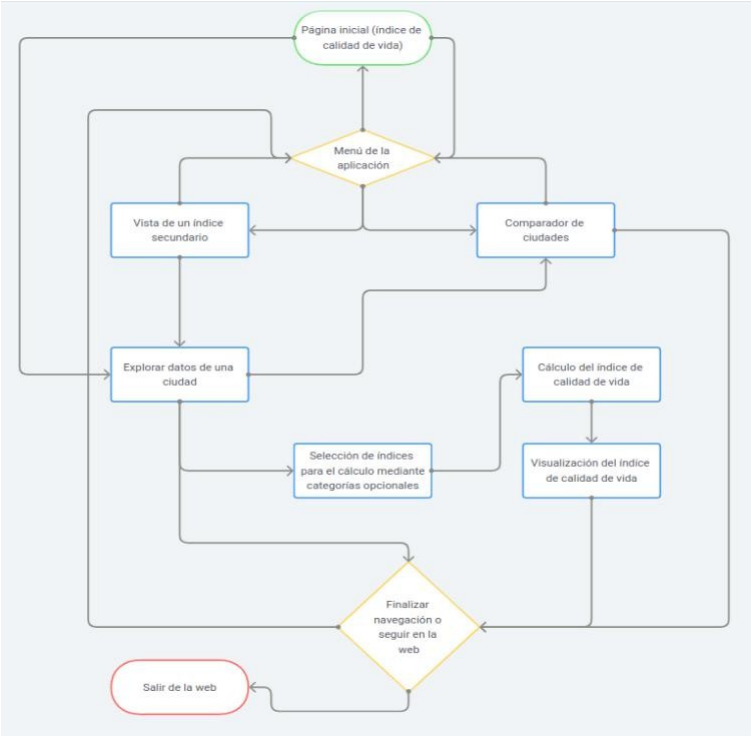


Figura 8.9: Diagrama de flujo de la aplicación

8.2.4 Prototipado Hi-Fi

El prototipado Hi-Fi o de alta fiabilidad, se usa para simular tanto el diseño de la aplicación, como la interacción que el usuario pueda realizar con esta. En este caso, el prototipado se ha realizado mediante la aplicación JustInMind y con un formato mobile first.

Este formato ayuda a optimizar mejor la distribución de los elementos, ya que al realizarse el diseño antes en el móvil que en el escritorio, se pueden observar las diferentes limitaciones a las que nos podemos enfrentar a posteriori.

-Versión Móvil[12]

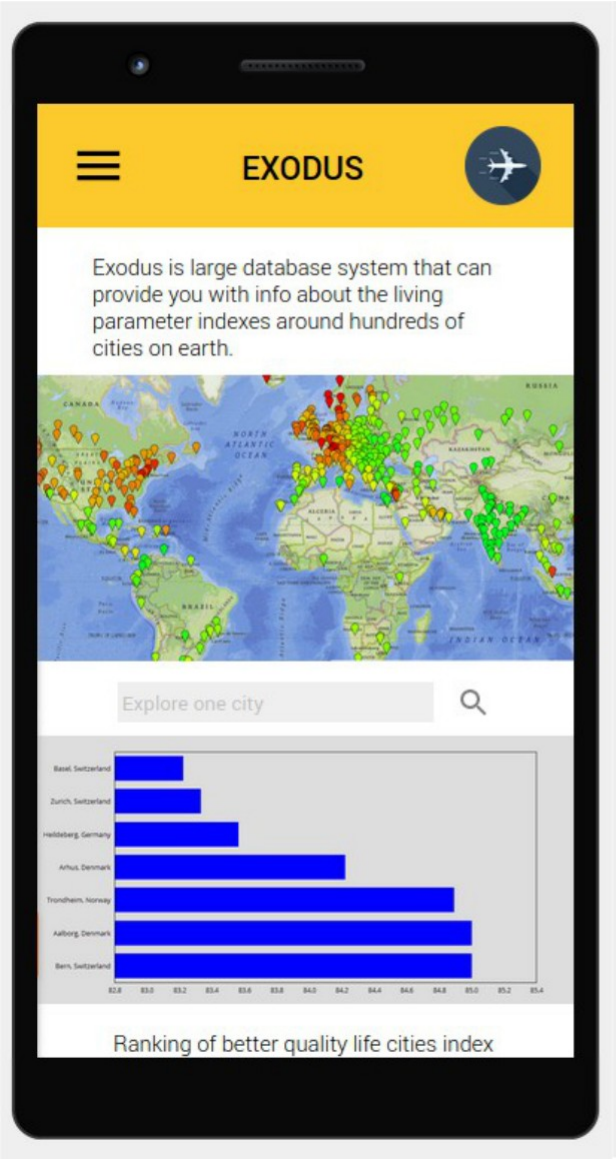


Figura 8.10: Pantalla de inicio de la aplicación

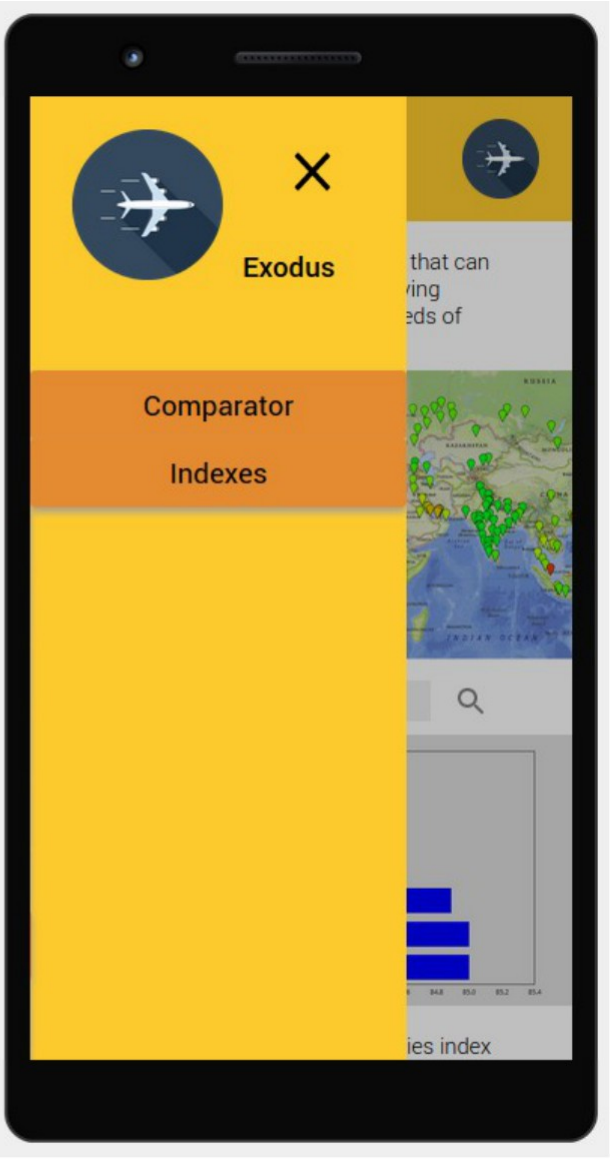


Figura 8.11: Menú condensado de la aplicación

12 La visualización del prototipo móvil puede verse en este enlace de la bibliografía.

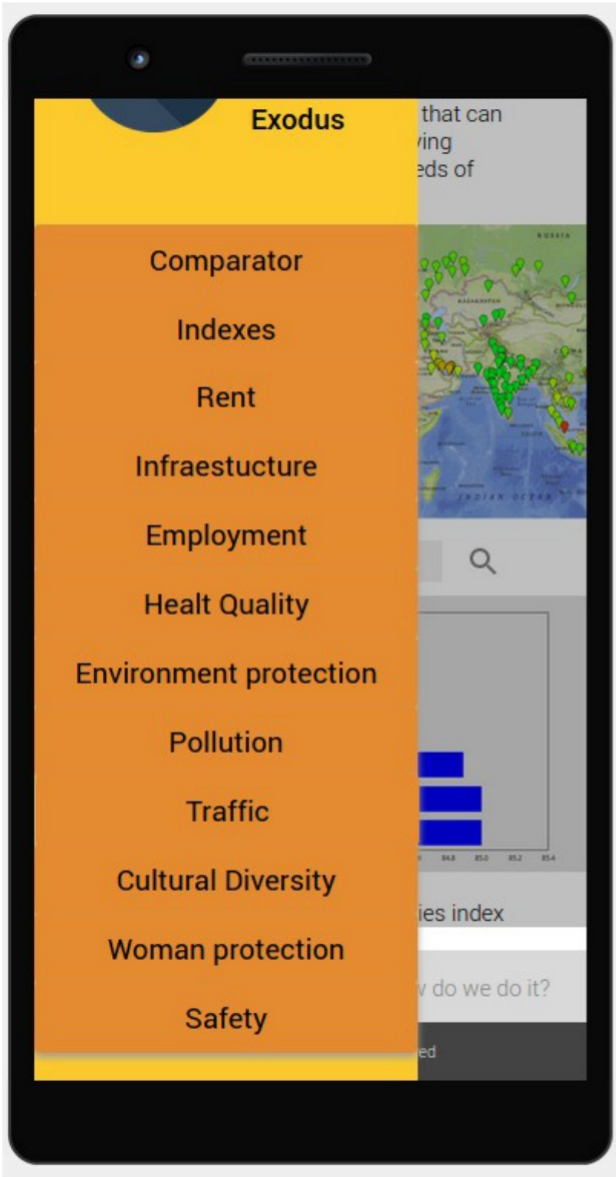


Figura 8.12: Menú extendido de la aplicación

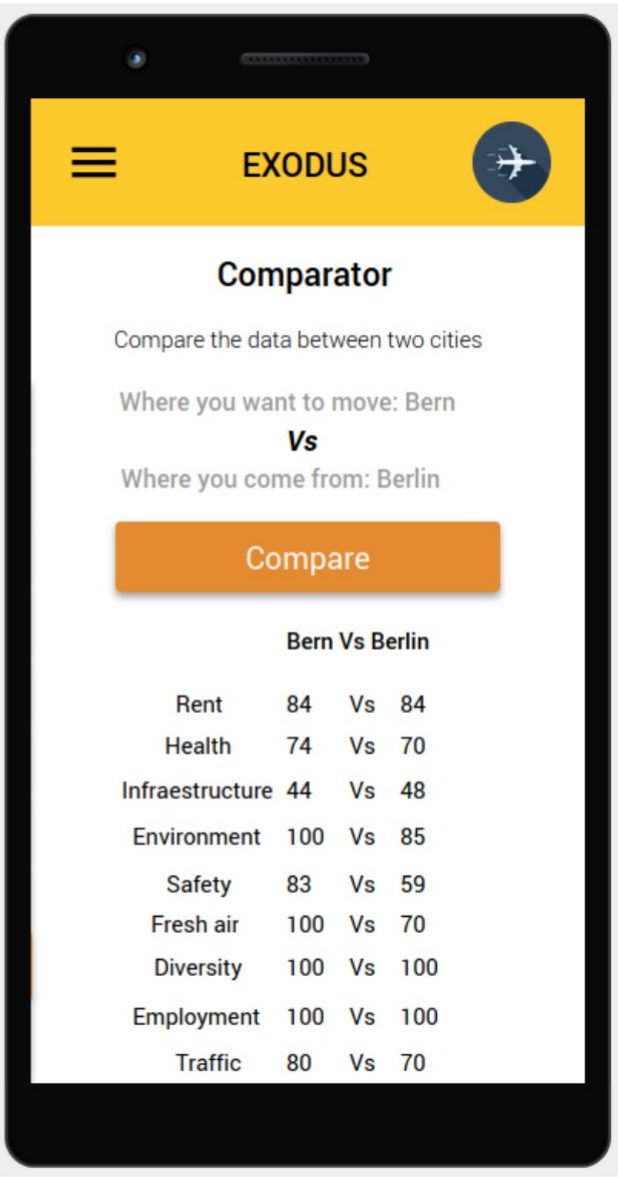


Figura 8.13: Comparador de la aplicación

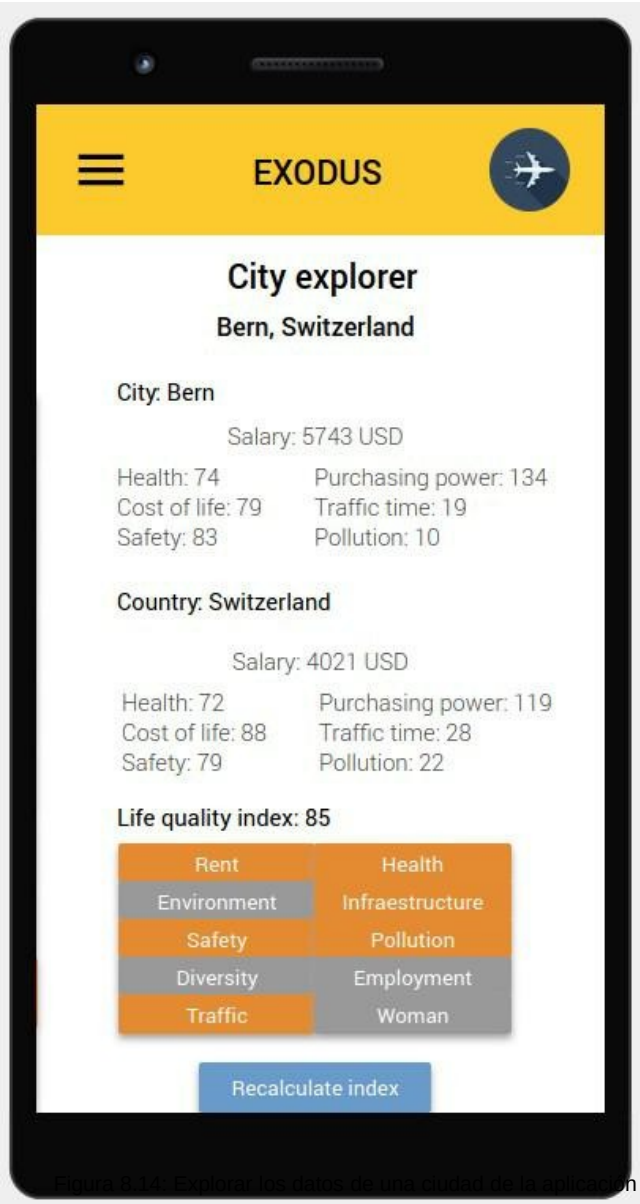


Figura 8.14. Explorar los datos de una ciudad de la aplicación

Como se puede observar, nuestra aplicación tendrá tres apartados fundamentales:

1- Los índices: La página inicial y las páginas de los subíndices, las cuales serán iguales, con la salvedad de la información que representen.

2- El comparador de ciudades, el cual establecerá una comparación de todos los índices de una ciudad.

3- El explorador de ciudades, en el cual, el usuario podrá visualizar los datos de una ciudad y los datos del país al que pertenece. Además tendrá disponibles los índices que forman el índice de calidad de vida, con el objetivo de que pueda seleccionar los que desee, para recalcular un nuevo índice en base a estos.

Además se dispondrá de un menú en el lateral izquierdo, con el que navegar por toda la web.

-Versión de escritorio[13]

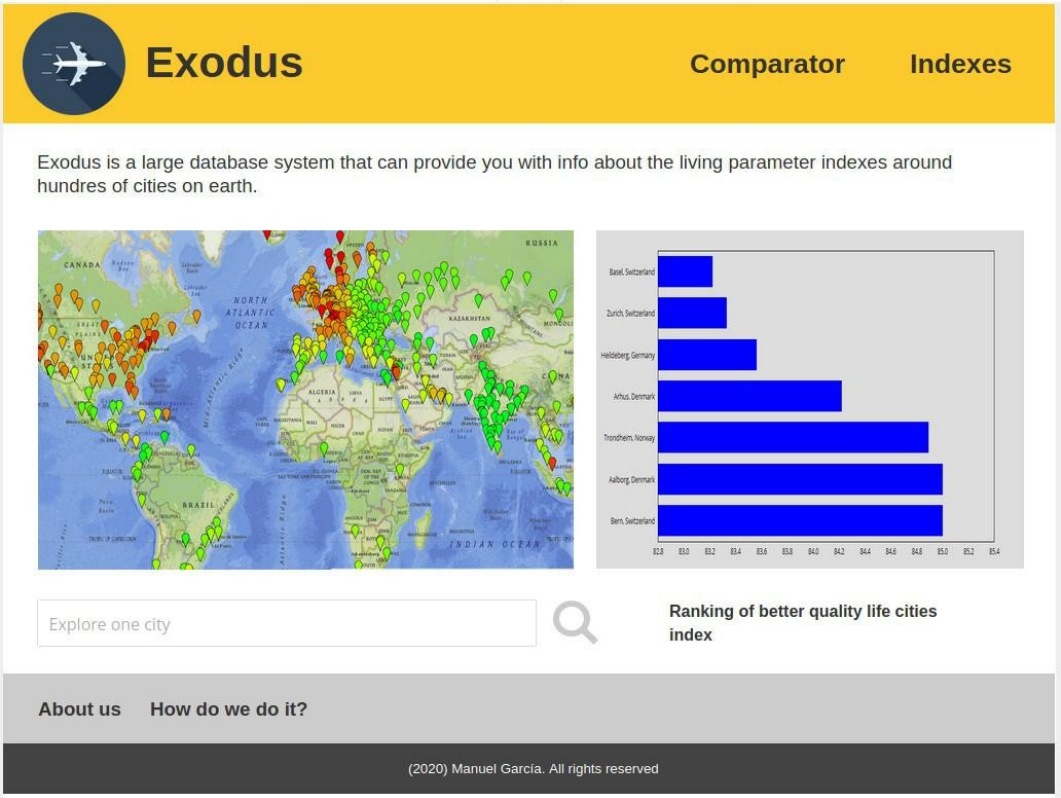


Figura 8.15: Página inicial en escritorio

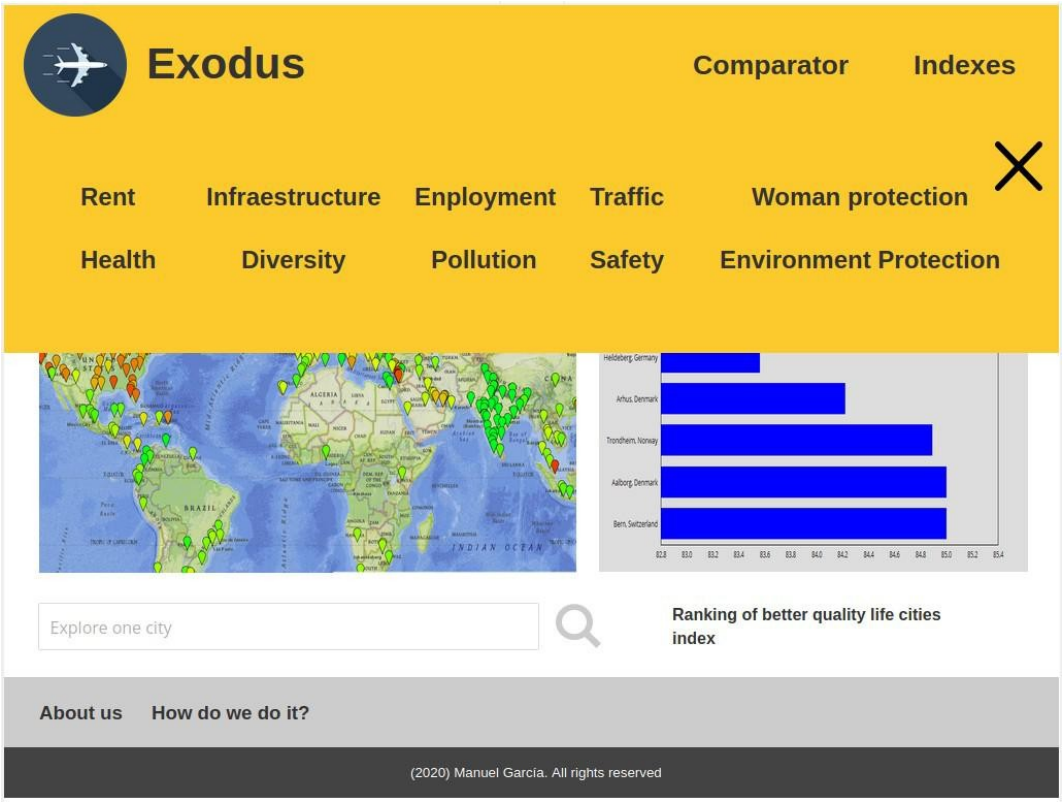


Figura 8.16: Menú de la aplicación en escritorio

13 La visualización del prototipo de escritorio puede verse en este enlace de la bibliografía.

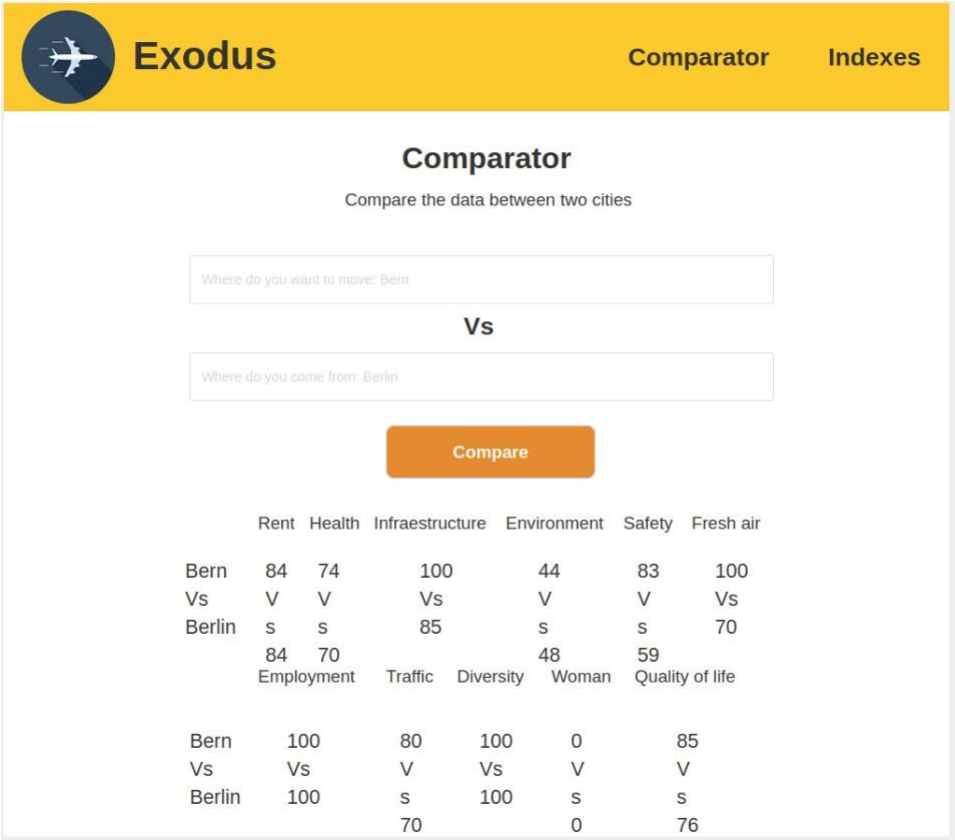


Figura 8.17: Comparador de la aplicación en versión escritorio

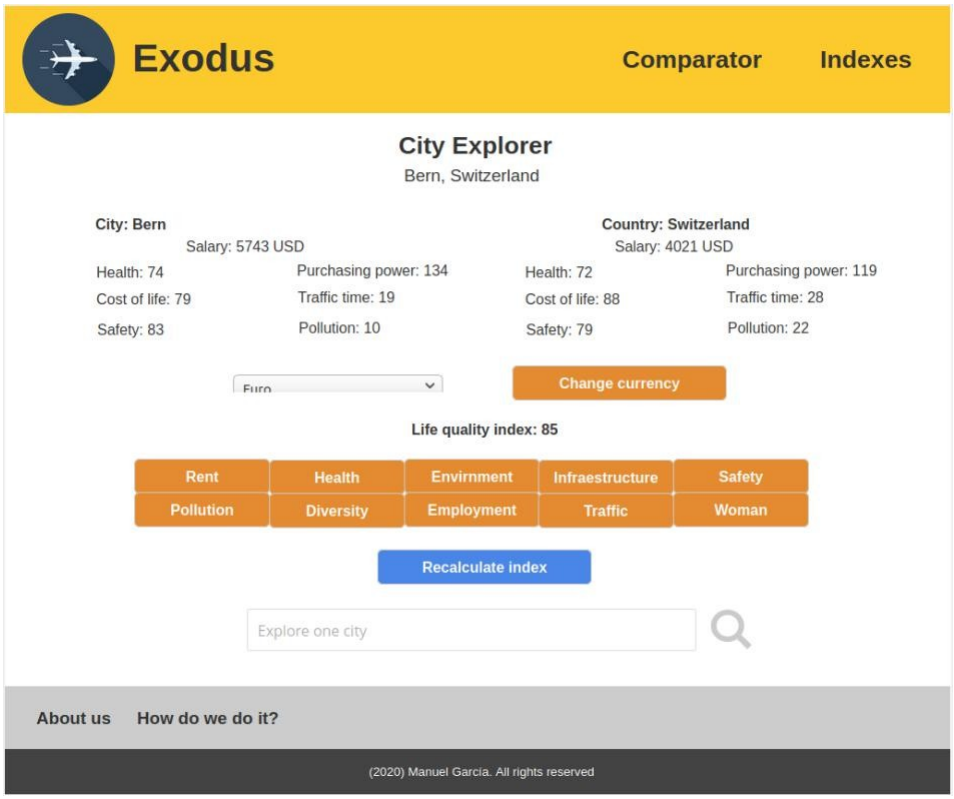


Figura 8.18: Datos de una ciudad de la aplicación en versión escritorio

8.3 Desarrollo e implementación de la aplicación

El desarrollo e implementación de la aplicación, se dividirá en tres fases fundamentales: Creación de la base de datos, creación del Back-end, y creación del Front-end.

8.3.1 Creación de la base de datos

La base de datos a utilizar es una base de datos de tipo sql. En concreto MariaDB, la cual es proporcionada por el gestor de bases de datos: phpmyadmin.

Una vez seleccionados, recogidos, filtrados y procesados los diferentes datos que compondrán nuestra aplicación, es el momento de crear las diferentes bases de datos que los almacenarán.

Para este proyecto, se contará con las siguientes bases de datos:

1) “Cities_data”: Donde se almacenarán los diferentes índices y datos de las ciudades.

2) “Countries_data”: Donde se almacenarán los diferentes índices y datos de los países.

3) “Salaries”: Donde se almacenarán además de los salarios de las ciudades, el índice de calidad de vida de cada una de ellas.

4) “World_cities”: Donde se almacenarán las diferentes ciudades que tenemos disponibles.

5) “Currencies”: Donde se almacenarán los valores de cambio de todas las monedas del mundo a euros y dólares estadounidenses. Los datos de esta tabla corresponden con la última actualización de Numbeo, el día en que se recogieron siendo este día, el 27 de Febrero de 2020.

El objetivo de almacenar la información de “Salaries”, es el de tener al alcance de la mano, la información de todas las ciudades que poseen un salario medio propio, ya que este es el criterio de filtro que vamos a seleccionar para dibujar dichas ciudades en el mapamundi.

El objetivo de almacenar “World cities”, es el de disponer de una tabla solo con las diferentes ciudades, sobre la que consultar a posteriori en el buscador de ciudades del front-end.

Además, encontraremos otras dos tablas: “api_tokens” y “migrations”.

6) “Api_tokens”: En esta tabla se almacenarán y actualizarán los accesos a la API del Back-end que no se produzcan desde el lado de la interfaz del cliente, sino mediante acceso directo vía url. El objetivo será controlar que no se produzcan accesos masivos de forma continua por parte de terceros, los cuales puedan ralentizar o bloquear el funcionamiento correcto de la aplicación.

7) “Migrations”: Esta tabla contiene un seguimiento de todas las tablas creadas en Back-end. Es creada de forma automática por el framework Laravel.

-SQL vs NoSQL

Llegados a este punto, quiero aclarar la elección de una base de datos de tipo SQL frente a otras de tipo NoQSL.

Si observamos la base de datos, veremos que las diferentes tablas no poseen una clave primaria de búsqueda, como cabría esperar de una base de datos de tipo SQL. Esto es así porque SQL nos permite realizar búsquedas por todas y cada una de las columnas de valores de cada tabla. Esto unido a que nuestros datos no poseen por lo general características únicas de clave primaria, nos facilita el uso de SQL.

Sin embargo, precisamente esas características son las que hacen de diversas bases de datos de tipo NoSQL como MongoDB, una alternativa excelente a SQL. Es aquí cuando entra en juego Laravel. Laravel, el cual usaremos de framework Back-end para construir nuestra API, incorpora un sistema de comunicación con lenguaje SQL muy sencillo y con una extensa documentación.

Es por esto, por lo que pese a disponer de sistemas de unión de bases de datos NoSQL como MongoDB con Laravel; se ha decidido utilizar SQL para esta versión del proyecto.

8.3.2 Creación del Back-end

El entorno Back-end se va a realizar con la ayuda del framework de desarrollo PHP, Laravel. Este entorno nos permitirá construir una API de forma muy sencilla, utilizando los datos de la base de datos que hemos creado anteriormente.

-Instalación de Laravel 6.0

Para trabajar, usaremos la versión seis de Laravel, para lo cual deberemos tener también instalado composer. Este será nuestro framework de trabajo para desarrollar y ejecutar el Back-end; el cual además enlazará con la base de datos de PhpMyAdmin.

-Modelos y migraciones

Una vez instalado y creado el proyecto, vamos a iniciar su configuración. Para conectar con la base de datos e interactuar con ella, realizaremos la configuración de conexión; y a continuación, las migraciones. Las migraciones son un protocolo de Laravel que nos permiten tanto crear las diferentes bases de datos, como enlazarlas entre sí por medio de los modelos de Laravel. Esto último no será necesario puesto que nuestras bases de datos no se relacionan entre sí en esta versión de la aplicación, por lo que dichos modelos permanecerán vacíos.

Para crear las migraciones y modelos, usaremos el comando “php artisan make:model (nombre del modelo) -m”. Una vez creadas y rellenadas todas las migraciones con los datos que debe poseer cada tabla, ejecutaremos el comando: “php artisan migrate”. Este comando creará en nuestra base de datos todas las tablas.

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateCountriesDataTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('countries_data', function (Blueprint $table) {
            $table->string('COUNTRY');
            $table->string('SALARY');
            $table->string('CPI');
            $table->string('PURCHASING_POWER');
            $table->string('POLLUTION');
            $table->string('SAFETY');
            $table->string('TRAFFIC');
            $table->string('HEALTH');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('countries_data');
    }
}
```

Figura 8.19: Ejemplo de una migración.

Una vez creadas todas las bases de datos, procederemos a rellenarlas. En nuestro caso, al disponer de datos reales, utilizaremos un script en lenguaje Python para introducir los valores ya procesados en cada una de las tablas de datos.

-Rutas y controladores

Para utilizar la información de nuestra base de datos, y construir así nuestra API, crearemos diversas funciones en un controlador al que llamaremos API, y accederemos a ellas mediante URLs desde un archivo de enrutamiento propio para APIs en laravel. Al tratarse de una API que consume recursos pero que no los genera, todas las rutas de nuestra API serán de tipo GET. Las diversas funciones de la API nos devolverán una respuesta en formato JSON si la petición ha sido exitosa, o en formato String si la petición ha fallado, en dicho string se especificará el motivo del fallo.

Veamos un ejemplo:

```php
// Cambios de moneda

public function allCurrencies($secret){

    $auth = true;

    if($secret != 'common_user'){
        $auth = $this->checkSecret($secret);
    }

    if($auth == true){
        $currencies = DB::select('select * from currencies');
        $data = array();
        for($i = 0; $i < sizeof($currencies); $i++){
            $info = $currencies[$i]->CURRENCY;
            array_push($data,$info);
        }
        return $data;
    } else {
        echo "Incorrect authentication token or request limit number reached";
    }
}
```

Figura 8.20: Ejemplo de función API.

```php
Route::get('/currencies/{secret}', 'ApiController@allCurrencies')->name('currency');

Route::get('/currencychange/{actualcurrency}/{newcurrency}/{secret}', 'ApiController@getCurrencyChange')->name('currencychange');

Route::get('/salaries/{index}/{secret}', 'ApiController@salaryCities')->name('salaries');

Route::get('/citydata/{city}/{country}/{secret}', 'ApiController@cityData')->name('cityData');

Route::get('/countrydata/{country}/{secret}', 'ApiController@countryData')->name('countryData');

Route::get('/worldcities/{text}/{secret}', 'ApiController@worldCities')->name('worldCities');

Route::get('/allcities/{secret}', 'ApiController@allCities')->name('allCities');
```

Figura 8.21: Ejemplo del archivo routes.

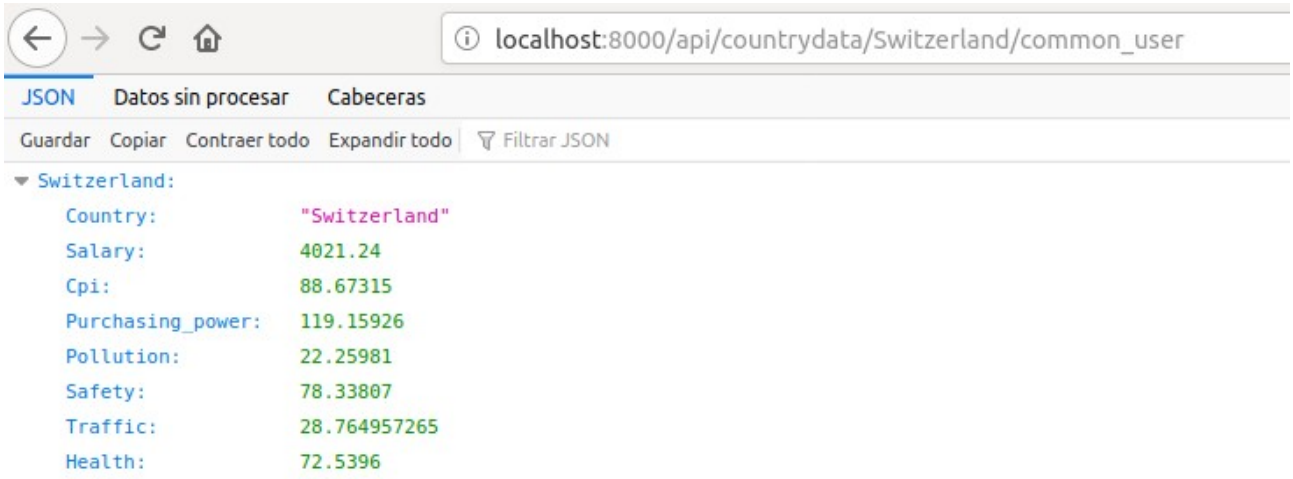


Figura 8.21: Ejemplo de respuesta de la API

8.3.3 Creación del Front-end

Una vez finalizada la implementación del back-end, es el momento de empezar a construir la aplicación del lado del cliente o front-end; dando a así uso a la información recopilada y la API recién creada.

- Componentes

La página está dividida en diversos componentes: El componente principal o app, y los componentes secundarios que desarrollan la lógica de la aplicación: Indexes, Comparator, Explorer, About y Api-info.

App

El componente App se encarga de iniciar la aplicación y de gestionar la información entre módulos. Además posee la estructura del menú de navegación, y del pie de página o footer; ambos presentes a lo largo de toda la aplicación.

Indexes

Indexes es el componente inicial de la aplicación. Este componente muestra una breve definición del objetivo de la aplicación, así como un mapamundi con la información sobre un índice concreto de los diez índices que maneja la aplicación; en este caso el índice de calidad de vida o QLI[14].

La información mostrada en el mapa, será la clasificación por color en función de la nota del índice consultado, de las ciudades registradas en la base de datos con salario medio. Dicho color será naranja o rojo si la nota es baja y verde si es alta.

Por último, este componente mostrará una gráfica con las diez ciudades del mundo que presenten los mejores datos en relación al índice consultado.

Comparator

Comparator es el componente encargado de establecer una comparación entre dos ciudades. Para ello, presenta dos input form en los que seleccionar dos ciudades de las disponibles en la base de datos. Una vez realizado esto, compara los resultados de los diversos índices entre sí.

Además, comparator ofrece la posibilidad de comparar el clima y de buscar trabajo mediante enlaces directos a las webs: weather-averages.co.uk y a linkedin.com.

Explorer

El componente explorer analiza en mayor detalle los datos relativos a una ciudad concreta, separando los datos de la propia ciudad y comparándolos con los datos obtenidos del país al que pertenece dicha ciudad.

14 QLI: Siglas en inglés de Quality Life Index.

Además te permite cambiar la divisa en la que se mide el salario, geolocalizar la ciudad en un mapa y calcular el valor de un índice de calidad de vida asociado a dicha ciudad y en base a los parámetros que conforman el índice de calidad de vida y que al usuario más le interesen.

Por último también te muestra la información relativa a los datos climáticos de una ciudad concreta gracias a un enlace directo a: timeanddate.com/weather/.

About

About es un componente que explica el objetivo del desarrollo de la aplicación, así como muestra un ejemplo de cálculo de uno de sus índices, y detalla las fuentes de las cuales se han extraído los datos utilizados.

Api-info

Api-info es un componente dedicado a explicar como se realiza la conexión de la aplicación front-end con la API back-end.

- APIs utilizadas

Además del uso de nuestra API desarrollada en back-end y de los datos de las fuentes mencionadas anteriormente; la aplicación hace uso de tres API más: Mapboxgl, Angular Material y ng2-charts.

Mapboxgl

Mapboxgl es una librería que nos permite hacer uso de mapas interactivos e integrables en cualquier plataforma web. Además, Mapbox nos permite añadir diversas capas de información al mapa, lo que nos ha permitido el desarrollo de markers y popups con los que representar por coordenadas a nuestras ciudades.

Angular material

Esta API ha sido utilizada para dar estilo y maquetar la aplicación web de una forma visual y atractiva de cara al usuario. Algunos de los elementos de Angular material utilizados han sido menú, botones, iconos, selectores o la información de despliegue en acordeón.

Ng2-charts

Ng2-charts nos permite dibujar en tiempo real y de forma totalmente responsive, casi cualquier tipo de gráficas. Gracias a Ng2-charts, podemos dibujar la gráfica de barras correspondiente a cada uno de los índices de la aplicación en nuestro componente Indexes.

9. Seguridad de la aplicación

9.1 Identificación y tratamiento de los riesgos de seguridad

Debido al tipo de aplicación, la cual no realiza actualizaciones, borrados o inserciones en la base de datos. La seguridad de la aplicación es relativamente sencilla, y consiste en: Limitación del número de peticiones, control del Cross site request forgery, y redirección de rutas.

Limitación del número de peticiones

El motivo de la realización de esta tarea es el de evitar una saturación intencionada o causada por un fallo en el número de peticiones diarias que puede realizar un usuario a la API. Este método consigue bloquear de forma efectiva los ataques por desbordamiento a la base de datos, aunque no evita la saturación de la propia API. Para controlar esto último y evitar así ataques DDOS o de inundación, se debe combinar este método con otro de DNS por parte del propio servidor.-

Cross site request forgery (CSRF)

Este ataque de tipo exploit malicioso es bastante perjudicial y suele afectar a la integridad de los datos y de la API backend. Para evitar este tipo de ataques, Laravel provee una librería específica, la cual también es la encargada de gestionar que métodos se pueden establecer sobre su API.

Redirección de rutas

Una mala redirección de rutas dentro de nuestra aplicación actual, puede no significar un error de seguridad, sin embargo en la mayoría de aplicaciones, redirigir de forma incorrecta puede suponer dejar espacios libres por los que acceder a los datos de una aplicación.

Como ya hemos comentado, esta aplicación debido a sus funcionalidades, no tiene riesgos importantes de seguridad, sin embargo, cualquier escalado y adición de funcionalidades extra que se le puedan hacer, puede conllevar la aparición de nuevos “agujeros” de seguridad, los cuales será muy importante identificar y tratar de forma adecuada y a tiempo.

9.2 Bugs

De momento solo se ha detectado un bug en la aplicación, el cual en principio no afecta a la seguridad de la aplicación, pero sí ligeramente al rendimiento de esta. Este bug se encuentra en el explorer, en el apartado del cálculo de índice de calidad de vida personal. Este bug hace que al realizar click en cualquiera de los botones que actúan como checkbox, la página se desplace hasta el inicio de la misma, como si de un ancla se tratase. El bug se aprecia más en pantallas pequeñas que en las grandes.

10. Puesta en marcha y análisis de resultados

La versión preparada de esta aplicación será una versión Beta. Las versiones Beta son aquellas que si bien han pasado de forma satisfactoria los test internos, propios de una versión Alfa; aún no se han enfrentado al análisis del público en un escenario real y abierto.

Durante la fase Beta, analizaremos hasta qué punto la aplicación satisface los requisitos de nuestros usuarios. Para ello analizaremos los siguientes parámetros:

1) Usabilidad
2) Diseño UI
3) Confiabilidad de los datos
4) Seguridad de la aplicación

Resultados extraídos al probar en local con algunos usuarios:

1- Usabilidad de la aplicación

La usabilidad de una aplicación se puede medir mediante diversos factores como:

La estructura del contenido.
Que el contenido se ajuste a lo esperado por el usuario.
Que la web sea interactiva.
Que no existan errores en la navegación inesperados.
Poseer un diseño limpio y claro.

En general, la aplicación construida se comporta correctamente y cumple con todos estos factores.

2- Diseño UI

El diseño es muy importante si se trata de una aplicación de usuario como en este caso. Es por ello que se deben mantener los patrones principales de programación de interfaces.

En el caso de esta aplicación, se cumplen un largo listado de patrones, los cuales garantizan el funcionamiento esperado de la aplicación.

3) Confiabilidad de los datos

En una aplicación de este tipo, que refleja de forma simple unos resultados con la intención de ayudar a un usuario en la toma de decisiones personales importantes. Los datos y el nivel de confianza en que estos representan la realidad es algo muy a tener en cuenta.

En el caso de esta aplicación, los datos han sido extraídos de fuentes oficiales y de prestigio a nivel mundial. Por lo que, unido a un manejo y agrupación de estos datos de tipo planificado, se puede decir que los datos de la aplicación ofrecen un alto nivel de garantías.

4) Seguridad de la aplicación

La seguridad de nuestros datos privados es de suma importancia.

En el caso de esta aplicación no se registran datos de los usuarios, estos son de acceso totalmente anónimo. Esto es así ya que en esta versión de desarrollo, no tenemos interés por registrar datos de uso particular. Por este motivo, la aplicación se puede considerar segura, al no exponer datos personales de ningún tipo.

11. Conclusiones

Conclusiones personales acerca del proyecto realizado, el proceso de trabajo y los resultados obtenidos.

El desarrollo de la aplicación Exodus, ha sido una gran aventura. La idea, que surgió como consecuencia de una inquietud personal, tenía la complicación de encontrar suficientes datos, como para permitirnos obtener unos resultados adecuados, que no divergieran en exceso de la realidad.

El haber podido desarrollar dicha idea gracias al trabajo de fin de máster, ha resultado muy satisfactoria, además de ser con ello, una oportunidad excelente para poner en práctica todos los conocimientos adoptados durante el desarrollo del máster, en un único proyecto.

Esta ha sido mi primera aplicación completa utilizando bases de datos, Laravel y Angular. Y como resultado estoy orgulloso de haber conseguido una web que considero fácil de usar y de comprender.

Sin embargo, Exodus se encuentra en una versión Beta, y dado que encuentro el proyecto bastante interesante, me encantaría seguir trabajando en él, añadiendo una mayor y mejor cantidad de datos, ciudades e índices, con el fin de aportar una mejor experiencia de usuario.

12. Recomendaciones futuras

12.1 Características a añadir

En relación a futuras mejoras dentro de todo proyecto, solemos tender a pensar en añadir funcionalidades de forma que el proyecto sea más atractivo y versátil.

Entre estas funcionalidades, podrían estar las siguientes:

1- Introducción de datos por parte de los usuarios y de forma anónima (mediante dirección IP por ejemplo).

2- Adición de mayores características a explorar y a comparar entre las diversas ciudades.

3- Añadir una mayor cantidad de índices de tipo general como los ya presentes en la web.

12.2 Características a mejorar

Sin embargo, no debemos olvidar, y menos en un proyecto de este tipo, el cual está basado en una gran cantidad de datos, que lo más importante es la calidad y variabilidad de los mismos. Por ello como ideas a futuro en cuanto a mejora de funcionalidades existentes, tenemos las siguientes:

1- Mejora de los datos almacenados, aportando un mayor número de datos por índice, y reemplazando aquellos que sea posible a nivel de país, por datos a nivel de ciudades.

2- Añadir los datos sobre salarios medios de muchas más ciudades, ya que este es uno de los datos disponibles más interesantes.

3- Actualizar todos los datos de forma periódica, por ejemplo, el cambio de divisas.

4- Dibujar una mayor cantidad de marcadores en el mapa según hagamos zoom en él.

5- Poder visualizar los datos de las mejores ciudades por sectores mas específicos, como continentes o regiones de continentes.

6- Poder comparar un mayor número de ciudades, al mismo tiempo.

12.3 Mejoras en el desarrollo

En cuanto al desarrollo, el posible escalado de la aplicación, puede hacer que mantener controlado el flujo de desarrollo sea cada vez más complicado. En cuanto a esto, existe la alternativa de adoptar otros patrones web como REDUX. Sin embargo, debido a que nuestra aplicación no va a manejar internamente datos cambiantes, no se considera que la adopción de REDUX pueda suponer un factor determinante para mejorar el desarrollo.

Algo que si que podría ser tenido en cuenta, es establecer una mayor modularización por medio de la adición de componentes a la web. Esto permitiría mantener el código mejor ordenado, lo cual ayuda a su comprensión.

12.4 Adaptación a formatos móviles

En cuanto a la adaptación a formatos móviles, existen dos enfoques principales. El salto de la web al mundo de las aplicaciones, o la mejora dentro exclusivamente del mundo web.

En principio, debido a que la web se interpreta más como una herramienta de ayuda a la toma de decisiones concretas que una web de uso continuado, lo mejor es realizar la mayor cantidad de mejoras sobre el formato web existente y adaptar esta web a todos los dispositivos móviles posibles mediante el uso de responsive design.

13. Requisitos de instalación y uso

-Requisitos hardware

En el lado del cliente:

Los requisitos hardware de la aplicación son mínimos, se podrá utilizar todo hardware capaz de ejecutar una pequeña aplicación web, y con memoria RAM suficiente como para almacenar una pequeña página web.

En el lado del servidor:

Los requisitos serán los habituales a manejar en toda API. Se requiere que el servidor se capaz de manejar una carga moderada de peticiones.

-Requisitos software

En el lado del cliente se requerirá un navegador actualizado.

En el lado del servidor se requerirá tener instalado, un gestor de bases de datos del tipo PHP como Xampp, PHP 7.2 o superior, Composer y Laravel con la versión 6 de Laravel, NodeJS con la versión 12.16.2 o superior y Angular cli y Angular core en las verisiones 9.0.7 o superior.

-Requisitos en conocimientos

En cuanto a los conocimientos, en el lado del cliente solo se requiere los conocimientos básicos para navegar por internet. Mientras que en el lado del servidor, se requieren los conocimientos mínimos de manejo de programas mediante consola de comandos o terminal.

14. Instrucciones de instalación

Información con pasos detallados acerca de cómo se debe instalar/implantar el servicio/aplicación. Las presentes instrucciones deben también acompañar al servicio/aplicación en un archivo contenido en sus directorios.

Partimos de un punto en que el administrador web dispone de un servidor (no configurado, similar a aamazon web services), y un dominio, pero estos no se encuentran configurados para soportar nuestra aplicación.

1-Conexión a la máquina y descarga del proyecto (se puede hacer vía scp, wget o git clone).

2-Identificación de sus apartados principales: Base de datos, Back-end y Front-end.

3-Base de datos

1) Instalación de un entorno tipo Xampp o similar, que disponga de una base de datos de tipo SQL compatible con php.
2) Una vez la xampp esté activo (sudo /opt/lampp/lampp start). Crear y configurar un entorno de bases de datos, y una base de datos vacía.

4-Back-end

1) Instalación de Composer y Laravel 6 (Puede requerir de numerosas dependencias de PHP, si este no está instalado o actualizado a la versión 7).

2) Conectar Laravel a la base de datos para poder realizar las peticiones. (Configuración del archivo .env, y migraciones).

3) Una vez conectados Back-end y base de datos. Importar la base de datos de nuestro proyecto en la base en la base de datos vacía que acabamos de crear. (La importación se debe realizar mediante el script de populado de la base de datos: populatedb.py, el cual para ejecutarse requiere de Python3).

5-Front-end

1) Realizar los cambios oportunos en las rutas de uso de la API. (Solo en caso de que el backend no esté ejecutándose en localhost:8000 o 127.0.0.1:8000).

2) Subida del proyecto en Angular compilado (subida de la carpeta Dist). Si el proyecto no funciona mostrando una pantalla en blanco, hay que acceder al index.html de nuestra carpeta Dist, y la etiqueta base href, cambiarla por: <base href="./">

6-Ejecución

1) Comprobar que el proyecto funciona correctamente

15. Manual de uso

Información con pasos detallados acerca de cómo se debe utilizar el servicio/aplicación. Las presentes instrucciones deben también acompañar al servicio/aplicación en un archivo contenido en sus directorios o como una función más del servicio/aplicación.

1- Apartado de Indexes.

En este apartado encontramos un mapamundi con las ciudades que disponen de salario medio en nuestra web. El mapa es interactivo, lo que supone que podemos navegar y realizar distintas acciones sobre él. Las acciones que podemos realizar sobre el mapa son: -Navegar por el mapa.

-Hacer zoom sobre cualquier parte del mapa.

-Hacer click sobre los marcadores del mapa, los cuales ejecutarán un pop-up que desplegará un pequeño apartado de información de dicha ciudad. Además nos permitirá navegar hasta la información correspondiente de dicha aplicación.

La información visualizada en el mapa, se rige por un código de color, el cual está determinado por la nota obtenida por cada una de las ciudades en cuanto al índice consultado. En este código de color, el rojo y el naranja representan notas negativas, el amarillo y amarillo-verdoso representan notas neutras y el verde claro y verde oscuro representan notas positivas.

Además del mapa posee un gráfico que nos permite visualizar las diez mejores ciudades del mundo en el aspecto relativo a un índice concreto, inicialmente el índice de calidad de vida.

Finalmente encontramos enlaces directos al resto de las secciones de la web, además de la opción de navegar mediante el menú principal.

2- Apartado de comparator.

En el apartado del comparador de ciudades, encontraremos dos buscadores de tipo input. Al escribir sobre ellos, podremos visualizar una serie de ciudades las cuales se irán filtrando a medida que vayamos añadiendo mas caracteres a la búsqueda. Para realizar correctamente una búsqueda se hará click en una de las ciudades que contengan el patrón de caracteres buscado.

Una vez se hayan elegido las dos ciudades a comparar, se tendrá que hacer click sobre el botón “compare cities” y se mostrará la comparativa entre ellas, a través de los distintos valores que se posean. Además también se mostrará información relativa al clima comparativo de ambas ciudades, salarios, coste de vida o poder adquisitivo.

También se permite buscar trabajo en alguna de dichas ciudades.

3-Apartado de explorar ciudades.

El apartado de explorar ciudades nos ofrece un buscador que funciona de forma similar al buscador del comparador. Para activar la búsqueda, haremos click en explore city una vez seleccionada la ciudad que deseemos.

En este apartado encontraremos una sección que nos permite seleccionar los diferentes índices, desde uno nueve, que componen el índice de calidad de vida; a fin de poder calcular un índice de calidad de vida para dicha ciudad, acorde a nuestros intereses personales. Para ello, haremos click en calculate personal index value una vez marcados todos los chekbox deseados.

También se nos muestra la opción de realizar un cambio de divisas, para ello una vez escogida la nueva divisa, solo tenemos que hacer click en change currencie, y esta actualizará el valor de los salarios medios al valor con la nueva divisa.

Por último encontramos información sobre la ciudad, el país, el clima y la posición exacta en un mapa de dicha ciudad, rodeada de sus ciudades vecinas.

4- About us

En este apartado podremos encontrar diversa información sobre la realización de la web, la motivación, y el cálculo de los datos representados.

5- Api

Finalmente, en el apartado de la API, se pueden observar las diferentes llamadas que se realizan al back-end y los datos que estas llamadas nos devuelven.

Anexo 1. Entregables del proyecto

Lista de archivos entregados y su descripción.

Archivo: **PAC_FINAL_García_López_Manuel.zip**

Contiene la entrega completa, es decir, la documentación, la presentación, el video de presentación y el proyecto.

Archivo: **PAC_FINAL_prj_García_López_Manuel**

Contiene el front-end, back-end y la base de datos

1. Exodus-frontend
2. Exodus-backend
3. Exodus-database

Archivo: **PAC_FINAL_mem_García_López_Manuel**

Contiene la memoria, los anexos y la autoevaluación.

1. Memoria completa del trabajo de fin de master

   PAC_FINAL_mem_García_López_Manuel.pdf

2. Anexos

   PAC_FINAL_competencias_transversales_del_proyecto_García_López_Manuel.pdf

Archivo: **PAC_FINAL_vid_García_López_Manuel**

Contiene el video de la presentación de la aplicación

PAC_FINAL_vid_García_López_Manuel.mp4

Archivo: **PAC_FINAL_prs_García_López_Manuel**

Contiene la presentación de la aplicación en pdf y PowerPoint (LibreOffice Impress)

PAC_FINAL_prs_García_López_Manuel.pdf

PAC_FINAL_prs_García_López_Manuel.odp

Anexo 2. Código fuente (extractos)

Back-end: Ejemplo de una función de la API, llamada a cambio de moneda o “currency change”:

```
// Valor de cambio entre dos monedas

public function getCurrencyChange($actualCurrency, $newCurrency, $secret){

    $auth = true;

    if($secret != 'common_user'){
        $auth = $this->checkSecret($secret);
    }

    if($auth == true){
        // Cambio a dolares de la moneda actual
        $USDChangeActual = DB::table('currencies')->where('CURRENCY', $actualCurrency)->first();
        if($USDChangeActual != null){
            $USDChangeActual = (float) $USDChangeActual->USD_TO_CURRENCY;

            // Cambio a dolares de la moneda deseada
            $USDChangeNew = DB::table('currencies')->where('CURRENCY', $newCurrency)->first();
            if($USDChangeNew != null){
                $USDChangeNew = (float) $USDChangeNew->USD_TO_CURRENCY;

                // Factor de cambio de la moneda acutal a la moneda deseada
                $change = $USDChangeNew/$USDChangeActual;

                //Devolver factor de cambio
                return $change;
            } else {
                echo "The new currency change does not exist";
            }

        } else {
            echo "The actual currency change does not exist";
        }

    } else {
        echo "Incorrect authentication token or request limit number reached";
    }
}
```

Front-end: Funciones consumidoras de nuestra Api back-end:

```
getMapCities(index): Observable<any[]> {
  const complete_api = this.GENERAL_ENDPOINT + '/salaries/'+ index +'/common_user';
  return this.httpClient.get<any>(complete_api);
}

getWorldCities(text): Observable<any[]> {
  const complete_api = this.GENERAL_ENDPOINT + '/worldcities/'+ text +'/common_user';
  return this.httpClient.get<any>(complete_api);
}

getAllCities(): Observable<any[]> {
  const complete_api = this.GENERAL_ENDPOINT + '/allcities/common_user';
  return this.httpClient.get<any>(complete_api);
}

getCityData(city, country): Observable<any[]> {
  const complete_api = this.GENERAL_ENDPOINT + '/citydata/' + city + '/' + country + '/common_user';
  return this.httpClient.get<any>(complete_api);
}

getCountryData(country): Observable<any[]> {
  const complete_api = this.GENERAL_ENDPOINT + '/countrydata/' + country + '/common_user';
  return this.httpClient.get<any>(complete_api);
}

getCurrencies(): Observable<any[]> {
  const complete_api = this.GENERAL_ENDPOINT + '/currencies/common_user';
  return this.httpClient.get<any>(complete_api);
}

getCurrencyChange(actualCurrency, newCurrency): Observable<number> {
  const complete_api = this.GENERAL_ENDPOINT + '/currencychange/' + actualCurrency + '/' + newCurrency + '/common_user';
  return this.httpClient.get<any>(complete_api);
}
```

Uso de dichas funciones en nuestra aplicación:

```
constructor(private router: Router, private actRoute: ActivatedRoute, private apiService: ApiService,
  private mapService: MapService, private chartService: ChartService) {
  this.barChartData = this.chartService.dataChart('Qli');
  this.barChartLabels = this.chartService.labelsChart('Qli');
  this.aux = this.actRoute.snapshot.params.id;
  this.router.routeReuseStrategy.shouldReuseRoute = function () {
    return false;
  };

  this.mySubscription = this.router.events.subscribe((event) => {
    if (event instanceof NavigationEnd) {
      this.router.navigated = false;
    }
  });
}

ngOnInit(): void {
  this.onMapChange();
}

onMapChange(): void {
  if(typeof this.aux === 'undefined'){
    this.index = 'Qli'
  } else {
    this.index = this.aux
  }
  document.getElementById('map').innerHTML="";
  this.apiService.getMapCities(this.index)
  .pipe(
    tap(data => this.mapService.buildMap(this.index, data, this.zoom)),
    tap(() => this.barChartData = this.chartService.dataChart(this.index)),
    tap(() => this.barChartLabels = this.chartService.labelsChart(this.index)),
  )
  .subscribe();
}

ngOnDestroy() {
  if (this.mySubscription) {
    this.mySubscription.unsubscribe();
  }
}
}
```

Anexo 3. Librerías/Código externo utilizado

Información detallada acerca de qué librerías, código, archivos, y cualquier otra herramienta tecnológica desarrollada por terceros utilizada en el trabajo, y qué partes de los mismos han sido usadas y cómo.

Mapboxgl

https://docs.mapbox.com/mapbox-gl-js/examples/

Los mapas presentados en la web se han realizado gracias a MapboxGl, una librería interactiva que nos ha permitido reflejar los datos sobre ella de forma cómoda y manteniendo el formato responsive.

Ng2-charts

https://valor-software.com/ng2-charts/

Ng2-charts es un módulo instalable de angular, que nos permite realizar gráficas en tiempo real y en formato responsive con los datos que les proporcionemos.

Angular material

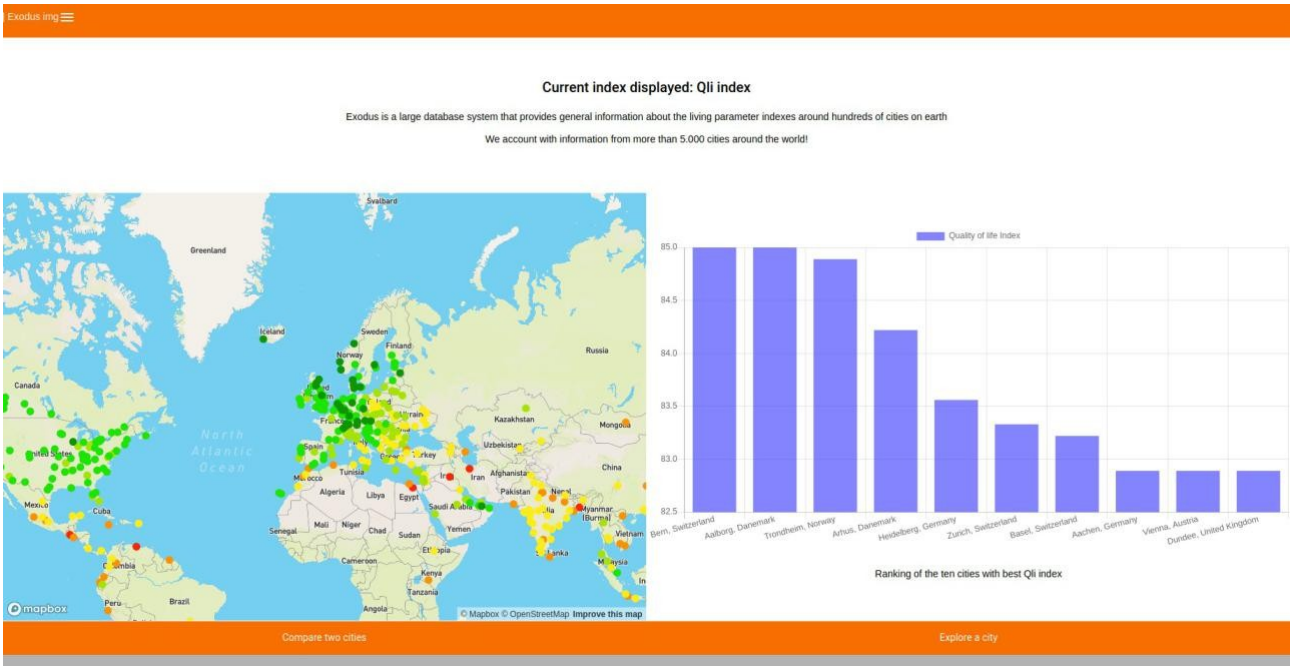
https://material.angular.io/components/categories

Angular material es una librería perteneciente a Angular que nos permite trabajar con elementos interactivos y responsivos ya diseñados bajo un estilo propio y moderno. Esta librería trabaja de forma similar a como lo harían otras tan famosas como Bootstrap, por lo que nos permite maquetar la aplicación de forma más cómoda y sencilla.

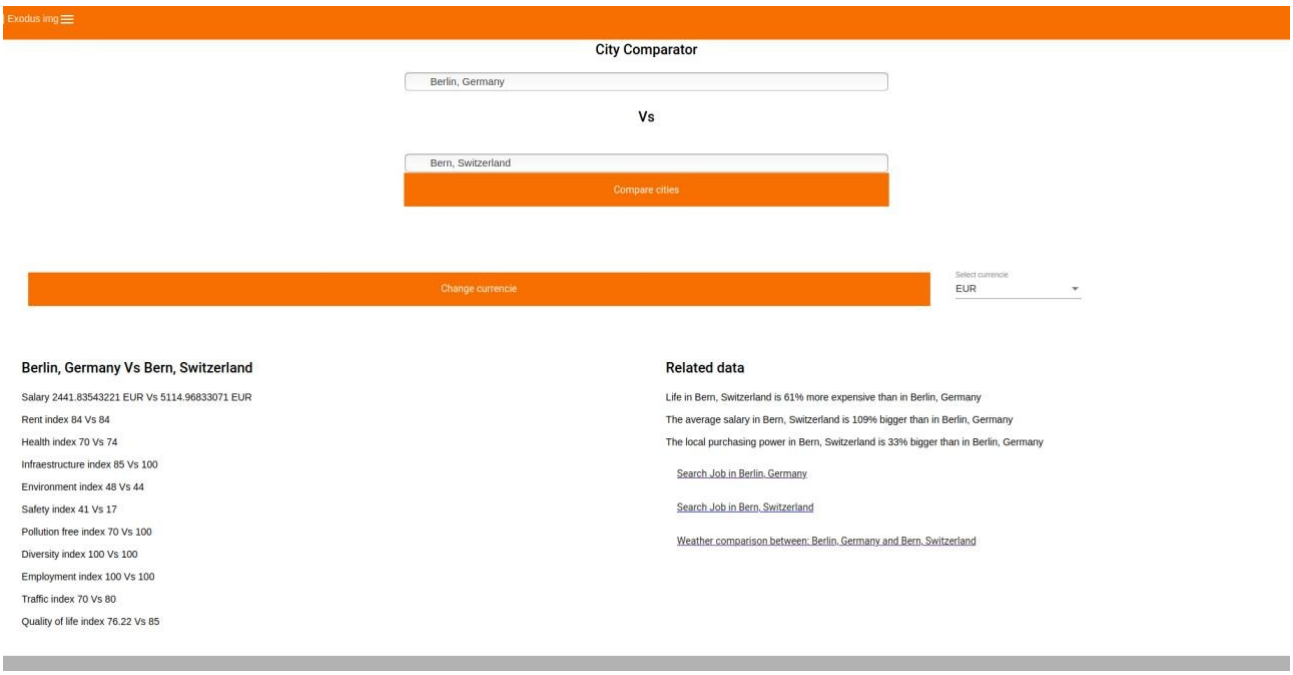
Anexo 4. Capturas de pantalla

Capturas de pantalla tanto del trabajo/servicio/aplicación realizado como del proceso de trabajo.

Pantalla principal

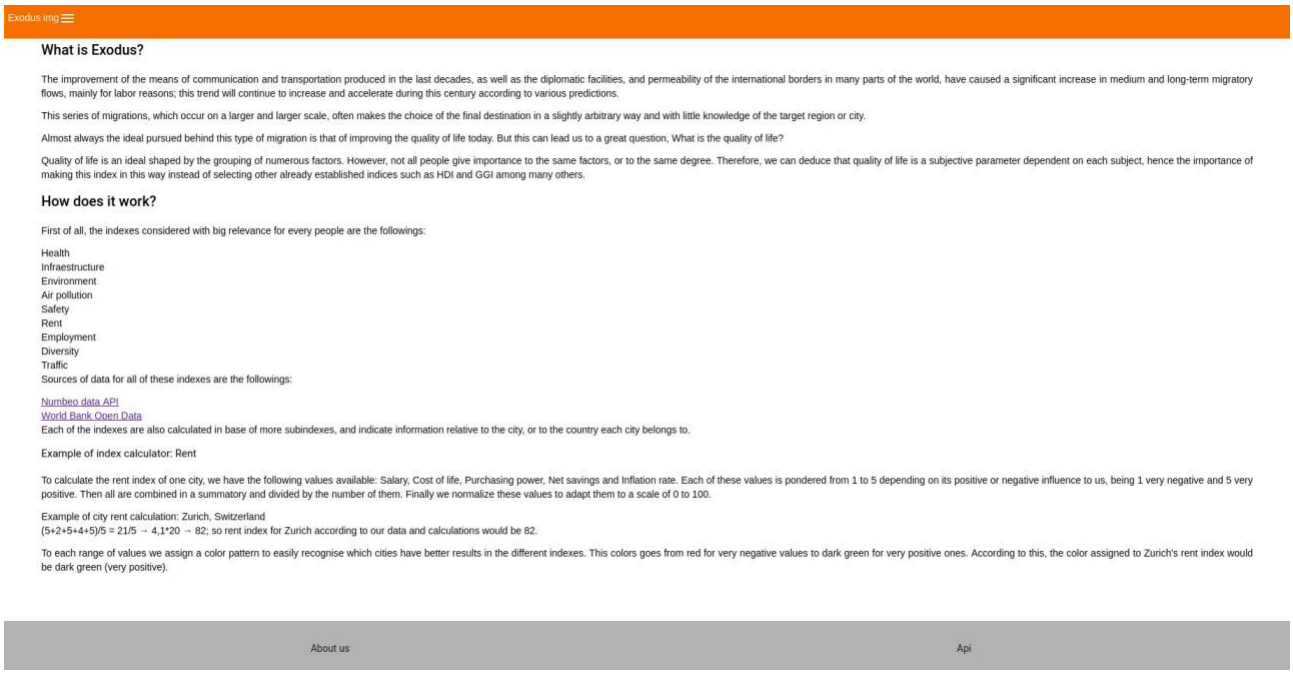


Pantalla comparativa entre dos ciudades

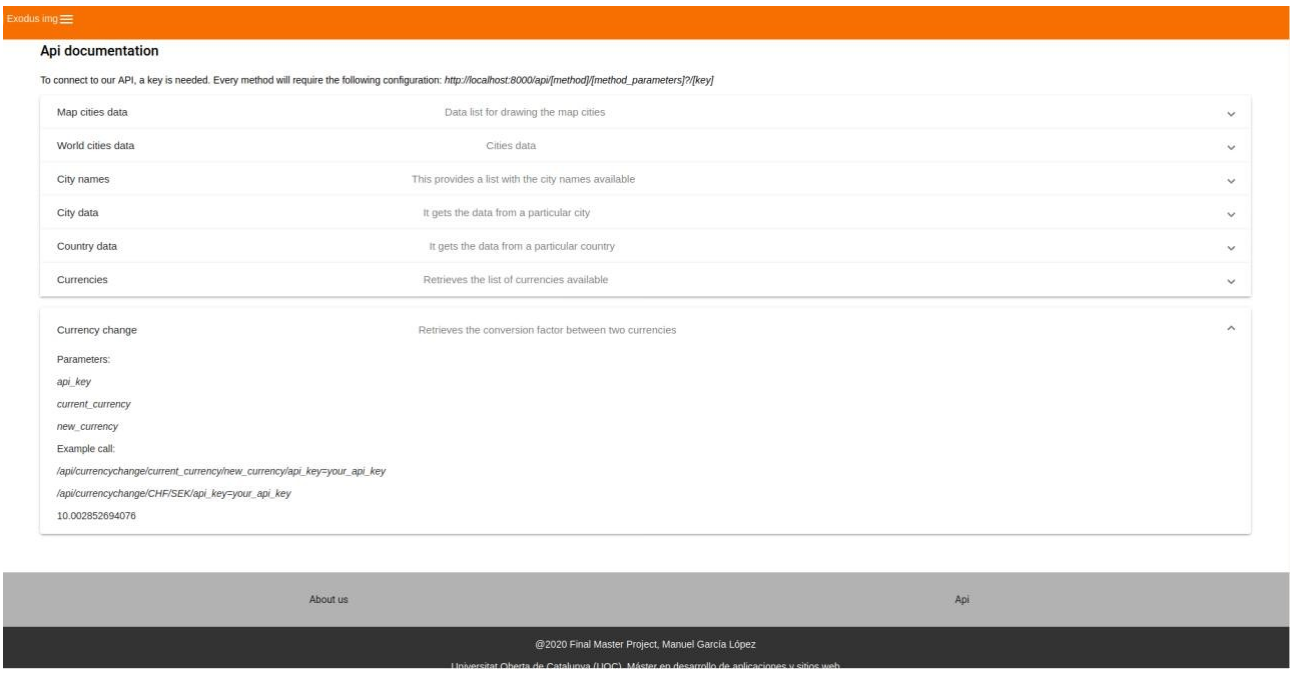


Pantalla explorar una ciudad

Pantalla About Us



Pantalla Api Information



Anexo 5. Bibliografía

Andrade, C. (2011). How to make a good Abstract for a scientific paper or conference presentation. *Indian Journal of Psychiatry.* [*PMC free article*]

Sudhir, A, Amartya, K. (1993). Human Development Index: Methodology and measurement. *Occasional papers. [PMC free arcticle]*

Migration forecasting (2020). Predicciones migratorias. https://migrationdataportal.org Obtenido de: https://migrationdataportal.org/themes/migration-forecasting

Good Governance Index (2019). Índice de buena governanza. https://en.wikipedia.org Obtenido de: https://en.wikipedia.org/wiki/World_Governance_Index

Calidad de vida (2020). Calidad de vida e índice de desarrollo humano. https://en.wikipedia.org Obtenido de: https://es.wikipedia.org/wiki/Calidad_de_vida

Numbeo (2020). Aplicación utilizada para realizar el análisis de mercado. https://www.numbeo.com/quality-of-life/ Obtenido de: https://www.numbeo.com/quality-of-life/

Nomad list (2020). Aplicación utilizada para realizar el análisis de mercado. https://nomadlist.com/ Obtenido de: https://nomadlist.com/

Expatistan (2020). Aplicación utilizada para realizar el análisis de mercado. https://www.expatistan.com/cost-of-living Obtenido de: https://www.expatistan.com/cost-of-living

World Bank Open Data (2020). Base de datos abiertos de todos los países del mundo. https://data.worldbank.org/ Obtenido de https://data.worldbank.org/

Google developers (2020). Responsive web patterns. https://developers.google.com/ Obtenido de: https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns

UI patterns (2020). User Interface patterns. https://ui-patterns.com Obtenido de: https://ui-patterns.com/patterns

Site map (2020). Árbol de navegación de la aplicación. https://app.flowmapp.com Obtenido de: https://app.flowmapp.com/share/1ae520d5b7671ccbef5c76f0618548a0/sitemap/

User flow (2020). User flow de la aplicación. https://app.flowmapp.com Obtenido de: https://app.flowmapp.com/share/1ae520d5b7671ccbef5c76f0618548a0/userflow/76947/

Mobile prototype JustInMind (2020). Repositorio privado de la UOC en JustInMind, visible con una cuenta en JustInMind. https://www.justinmind.com Obtenido de: https://www.justinmind.com/usernote/listPrototypes.action?projectId=44069062

Desktop prototype JustInMind (2020). Repositorio privado de la UOC en JustInMind, visible con una cuenta en JustInMind. https://www.justinmind.com Obtenido de: https://www.justinmind.com/usernote/prototypes/42119839/44069062/45844778/index.html#/screens/d12245cc-1680-458d-89dd-4f0d7fb22724

Laravel documentation (2020). Documentación oficial de Laravel 6.x. https://laravel.com/ Obtenido de: https://laravel.com/docs/6.x

Angular documentation (2020). Documentación oficial de Angular 9.x. https://angular.io/ Obtenido de: https://angular.io/

Angular material documentation (2020). Documentación oficial de Angular material 9.2.x. https://material.angular.io/ Obtenido de: https://material.angular.io/

Ng2-charts documentation (2020). Documentación oficial de Ng2-chart https://medium.com/codingthesmartway-com-blog/ Obtenido de: https://medium.com/codingthesmartway-com-blog/angular-chart-js-with-ng2-charts-e21c8262777f

MapboxGl documentation (2020). Documentación oficial de MapboxGl https://docs.mapbox.com/ Obtenido de: https://docs.mapbox.com/mapbox-gl-js/examples/

Anexo 6. Vita

Manuel García López, natural de Jaén y residente en la misma localidad. Es Ingeniero en Informática por la universidad de Jaén, finalizando los estudios en el año 2019.

En el curso 2017/2018, trabajó para la universidad como administrador de informática en la facultad de salud de la universidad de Jaén. Y en el curso 2018-2019, como investigador informático en el grupo SINAI de la escuela politécnica superior de la misma universidad.

En 2019 comenzó a trabajar en forma de voluntariado, como webmáster de la asociación Erasmus Student Network de Jaén.

Siempre interesado en el mundo web y de internet, comienza este máster en el año 2019.