

APP GESTICOLMENAR

JOSÉ VICENTE FALCÓ MILLA
DESARROLLO DE APLICACIONES WEB
2º CURSO - MARZO/JUNIO



ÍNDICE

| | |
|---|----|
| Agradecimientos..... | 6 |
| Resumen..... | 6 |
| Introducción..... | 7 |
| Estudio de viabilidad..... | 8 |
| Análisis DAFO..... | 8 |
| Estudio de mercado..... | 10 |
| Viabilidad del proyecto..... | 10 |
| Monetizar la aplicación..... | 12 |
| Análisis de requisitos..... | 13 |
| Descripción de requisitos..... | 13 |
| Caso de uso general..... | 15 |
| Diagrama Entidad-Relación..... | 16 |
| Paso a tablas..... | 17 |
| Users..... | 17 |
| Places..... | 18 |
| Apiaries..... | 19 |
| Queens..... | 21 |
| Beehives..... | 23 |
| Products..... | 24 |
| Diseases..... | 25 |
| Diagrama físico..... | 26 |
| Diagrama de clases..... | 27 |
| Diagrama de secuencias..... | 28 |
| Diagrama de actividades..... | 29 |
| Site Map..... | 30 |
| Mockups Desktop..... | 31 |
| Login..... | 31 |
| Registro..... | 31 |
| Vista principal/Colmenares..... | 32 |
| Vista Colmenares Modal Borrar..... | 32 |
| Vista Colmenas..... | 33 |
| Vista Ubicaciones..... | 33 |
| Vista Ubicaciones Modal Borrar..... | 34 |
| Vista Reinas..... | 34 |
| Vista distintas Gráficas..... | 35 |
| Vista detalles Colmena..... | 35 |
| Vista Colmena Modal Añadir Producto..... | 36 |
| Vista detalles Usuario..... | 36 |
| Vista Modal Borrar Usuario..... | 37 |
| Vista detalles Ubicación..... | 37 |
| Vista Formularios (edición/creación)..... | 38 |
| Vista Tareas Pendientes TODO..... | 38 |
| Mockups Móvil..... | 39 |
| Vista Login..... | 39 |
| Vista Registro..... | 39 |
| Vista menú desplegado..... | 40 |
| Vista principal/Colmenares..... | 40 |

| | |
|---|-----------|
| Vista Colmenares Modal Borrar..... | 41 |
| Vista Colmenas..... | 41 |
| Vista Ubicaciones..... | 42 |
| Vista Ubicaciones Modal Borrar..... | 42 |
| Vista Reinas..... | 43 |
| Vista distintas Gráficas..... | 43 |
| Vista detalles Colmena..... | 44 |
| Vista Colmena Modal Añadir Producto..... | 44 |
| Vista detalles Usuario..... | 45 |
| Vista Modal Borrar Usuario..... | 45 |
| Vista detalles Ubicación..... | 46 |
| Vista Formularios (edición/creación)..... | 46 |
| Vista Tareas Pendientes TODO..... | 47 |
| Codificación..... | 48 |
| Tecnologías elegidas y su justificación..... | 48 |
| Entorno servidor..... | 49 |
| Entorno cliente..... | 49 |
| Despliegue..... | 50 |
| Descripción de la instalación..... | 50 |
| El uso de CDNs..... | 51 |
| Despliegue hosting gratuito 000webhost..... | 52 |
| Herramienta de apoyo Git y GitHub..... | 55 |
| Conclusiones..... | 56 |
| Conclusiones sobre el trabajo realizado..... | 56 |
| Conclusiones personales..... | 57 |
| Posibles ampliaciones..... | 58 |
| Bibliografía..... | 59 |
| Libros, artículos y apuntes..... | 59 |
| Direcciones web..... | 59 |

AGRADECIMIENTOS

A la razón de todo esto, a mi fiel amigo, Noble.

RESUMEN

Aplicación Web destinada al manejo de la información relacionada con la apicultura. Poder almacenar, gestionar y visualizar los datos que permitan la correcta gestión, control y administración de los distintos colmenares pertenecientes a un apicultor. Aplicación desarrollada con el framework Laravel (PHP) junto con otras tecnologías tales como SQL, Bootstrap, HTML, CSS .

Aplicació Web destinada a tractar la informació relacionada amb l'apicultura. Poder emmagatzemar, gestionar i visualitzar les dades que permetin la correcta gestió, control i administració dels diferents ruscs pertanyents a un apicultor. Aplicació desenvolupada amb el framework Laravel (PHP) juntament amb altres tecnologies com SQL, Bootstrap, HTML, CSS.

Web application for managing information related to beekeeping. Being able to store, manage and visualize the data that allows the correct management, control and administration of the different apiaries belonging to a beekeeper. Application developed with the (PHP) Laravel framework together with other technologies such as SQL, Bootstrap, HTML, CSS.

INTRODUCCIÓN

Comenzaré la introducción con una pregunta retórica... ¿cuán importante crees que es la vida de las abejas en el planeta tierra?

Sí en efecto, son vitales, sin más. Así mismo, la apicultura, además produce una serie de productos beneficiosos para el ser humano, tales como la miel, edulcorante milenario, el polen y la apitoxina entre otros.

Bueno, pues vamos a explicar el porqué se realiza una aplicación web en la que cualquier apicultor pueda tener de forma ordenada y de gestión sencilla, los datos necesarios más importantes que pueda necesitar tener a mano, consultar y/o almacenar.

Con la App Gesticolmenar, el apicultor podrá tener la información más relevante de sus colmenares, de sus colmenas, datos sobre sus abejas reina, enfermedades activas en la colmena, los datos de las ubicaciones y recordatorios de actividades y tratamientos a llevar a cabo.

Toda esta información, organizada, visual y accesible, hace que, podamos llevar en nuestro bolsillo todo, que podamos modificarla in-situ, que podamos obtener gráficas que nos den un punto de vista más visual de ciertos datos, que no tengamos que rebuscar datos de referencias catastrales para otras gestiones o visitas al apiario de autoridades, por ejemplo.

El ahorro en de tiempo y la ausencia de pérdida de datos entre libretas y papeles varios que son el día a día habitual de la práctica totalidad de los apicultores, hace que trabajar con Gesticolmenar, por fin, haga que puedan seguir con sus actividades favoritas sin tener que gestionar archivadores, libretas y anotaciones por doquier.

ESTUDIO DE VIABILIDAD

Análisis DAFO

Fortalezas:

- Gesticolmenar ofrece una solución para los apicultores, ya que les permite gestionar y llevar un seguimiento de sus colmenares y colmenas de una manera más eficiente y fácil.
- Ofrece la posibilidad de visualizar gráficas que permiten analizar la producción de miel, polen o apitoxina.
- La plataforma es accesible desde cualquier lugar con conexión a internet, lo que permite a los usuarios gestionar sus colmenares y colmenas de forma remota.
- Una aplicación visualmente sencilla, poco cargada y a su vez con toda la funcionalidad necesaria hacen que una vez se empieza a usar, sea tan rápido introducir datos y manejarla, que se mantenga su uso.

Debilidades:

- La aplicación web de gestión de colmenares ya tiene cierta competencia en el mercado, por lo que hay que esforzarte en ofrecer una propuesta única y diferenciada para destacar, como las notificaciones de tareas pendientes.
- El mercado de la apicultura es relativamente pequeño y segmentado, lo que puede limitar el crecimiento potencial de la aplicación.
- Los apicultores en su gran mayoría suelen ser gente de cierta edad o jóvenes que han heredado tanto las costumbres como las ideas y procedimientos, estando la tecnología de la información bastante alejada de su día a día, por lo que, existe cierta reticencia a pasar del papel a las app.

- La falta de experiencia en el desarrollo de aplicaciones web puede afectar a la calidad y funcionalidad de la aplicación.

Oportunidades:

- La preocupación por el medio ambiente y la apicultura en particular ha ido en aumento en los últimos años, lo que podría generar una mayor demanda de soluciones digitales para la gestión de colmenares.
- La posibilidad de expandir el negocio hacia otros sectores relacionados con la apicultura, como la venta de productos derivados de la miel, como cremas, jabones y otros productos de cuidado personal.
- Generar ventas cruzadas de productos para el cuidado y mantenimiento de colmenares, desde las colmenas hasta los productos sanitarios y la alimentación

Amenazas:

- La competencia en el mercado puede limitar la adopción de tu aplicación por parte de los usuarios.
- La estacionalidad de la actividad apícola puede generar un impacto en los ingresos del negocio, ya que, puede tener períodos de inactividad.
- La posibilidad de que los apicultores no quieran almacenar cierta información en una aplicación, es una posible causa de reticencia al uso.
- La extinción masiva de las abejas pueden afectar a largo plazo al uso de la aplicación.

Estudio de mercado

El mercado de la apicultura está compuesto principalmente por pequeños productores y empresas dedicadas a la producción de miel y otros productos apícolas. Existen algunas soluciones digitales para la gestión de colmenares, pero son pocas y generalmente de lengua extranjera, enrevesadas y poco prácticas para el día a día. Hay una oportunidad para ofrecer una solución más accesible y fácil de usar para los pequeños productores y apicultores aficionados.

Gesticolmenar puede ser atractiva para los apicultores, ya que les permite gestionar y llevar un seguimiento de sus colmenares y colmenas de una manera más eficiente y fácil, y ofrecer una mayor transparencia sobre la producción. Para llegar a este mercado, será importante tener una estrategia de marketing adecuada, que permita dar a conocer la aplicación y llegar a los posibles clientes.

Viabilidad del proyecto

Recursos Hardware:

Para alojar la aplicación web, se necesitará un servidor web. El tipo de servidor dependerá del número de usuarios que se esperan y de la cantidad de datos que se manejen. En un principio, podremos utilizar servicios de alojamiento compartido, pero a medida que crezca la base de usuarios, será necesario invertir en un servidor dedicado o en servicios de nube para garantizar la escalabilidad y el rendimiento. En cualquier caso, ninguna de estas soluciones suponen de una inversión grande.

Recursos Software:

Laravel es un framework de PHP que ayuda a acelerar el desarrollo de la aplicación al proporcionar herramientas y características útiles para el desarrollo web, como autenticación, enrutamiento, manejo de sesiones, etc.

Bootstrap es una biblioteca de CSS y JavaScript que se utiliza para el diseño de la interfaz de usuario. Proporciona una variedad de componentes de interfaz de

usuario preconstruidos y personalizables que ayudan a crear una aplicación web atractiva y fácil de usar.

Base de datos relacional SQL, para almacenar y recuperar datos de la aplicación, se necesita una base de datos, en este caso, relacional.

Servidor web, para alojar la aplicación, se necesita un servidor web. Se podría utilizar Apache o Nginx, por ejemplo, junto con PHP y MySQL.

Es importante tener en cuenta que estos son solo algunos de los recursos de software necesarios para desarrollar la aplicación web. El proyecto puede requerir herramientas y tecnologías adicionales en función de sus necesidades y objetivos específicos a corto, medio o largo plazo.

Recursos humanos:

Para el desarrollo de la aplicación web de gestión de colmenares, se ha necesitado de un único desarrollador, pero, esto ha dejado patente bastantes carencias en el diseño gráfico de la misma. Así mismo, llegado el momento, se deberá poder atender también a los usuarios con dudas o problemas técnicos que puedan surgir. Así mismo, la comercialización y exposición de la aplicación al mercado, deberá ser gestionada por un buen marketing.

Si es cierto que inicialmente podría encargarme de todas las tareas a sabiendas de las deficiencias, es cierto que conforme la aplicación empiece a establecerse en el mercado, se deberán incluir nuevos roles para el correcto funcionamiento, escalabilidad del producto y cubrir las carencias.

Viabilidad temporal:

En cuanto a la viabilidad temporal del proyecto, es importante tener en cuenta la estacionalidad de la actividad apícola y la necesidad de ofrecer una solución atractiva y funcional para los usuarios. Es posible que la demanda de la aplicación varíe a lo largo del año, y que sea necesario adaptar la oferta y la estrategia de marketing para atraer a nuevos usuarios y fidelizar a los ya existentes. Es importante estar atento a las tendencias del mercado y a las necesidades de los usuarios para mantener la relevancia de la aplicación a largo plazo.

Monetizar la aplicación

Existen varias maneras de monetizar una aplicación, pero en el caso de Gesticolmenar, lo más importante es encontrar una estrategia de monetización que no requiera una gran inversión inicial o un alto costo de mantenimiento. Algunas opciones podrían ser:

Publicidad: La inserción de publicidad en la aplicación podría generar ingresos a través de anuncios en línea, pero es importante tener en cuenta que esto podría afectar la experiencia del usuario.

Modelo de suscripción: Podría establecerse un modelo de suscripción para acceder a funciones premium en la aplicación, como el acceso a estadísticas detalladas o herramientas avanzadas de análisis. Es importante asegurarse de que las funciones premium sean lo suficientemente atractivas como para que los usuarios estén dispuestos a pagar por ellas.

Venta de datos: En este modelo de monetización, la aplicación podría recopilar datos sobre la actividad de los usuarios y venderlos a terceros, como empresas de investigación de mercado. Sin embargo, es importante asegurarse de que se cumplan todas las regulaciones de privacidad de datos.

Venta de productos apícolas: La aplicación podría establecer acuerdos de colaboración con tiendas de productos apícolas y recibir una comisión por cada venta realizada a través de la aplicación.

Cualquiera de estas opciones podría ser viable para monetizar la aplicación, pero es importante analizar cuidadosamente cuál es la más adecuada, pudiendo ser una combinación de varias o de todas ellas.

ANÁLISIS DE REQUISITOS

Descripción de requisitos

Es importante tener en cuenta los requisitos y funcionalidades que deben ser implementadas en la aplicación para satisfacer las necesidades de un apicultor. Por ello, se han tenido en cuenta los aspectos más relevantes del trabajo diario en dicha profesión y han sido analizados e incluidos en el desarrollo.

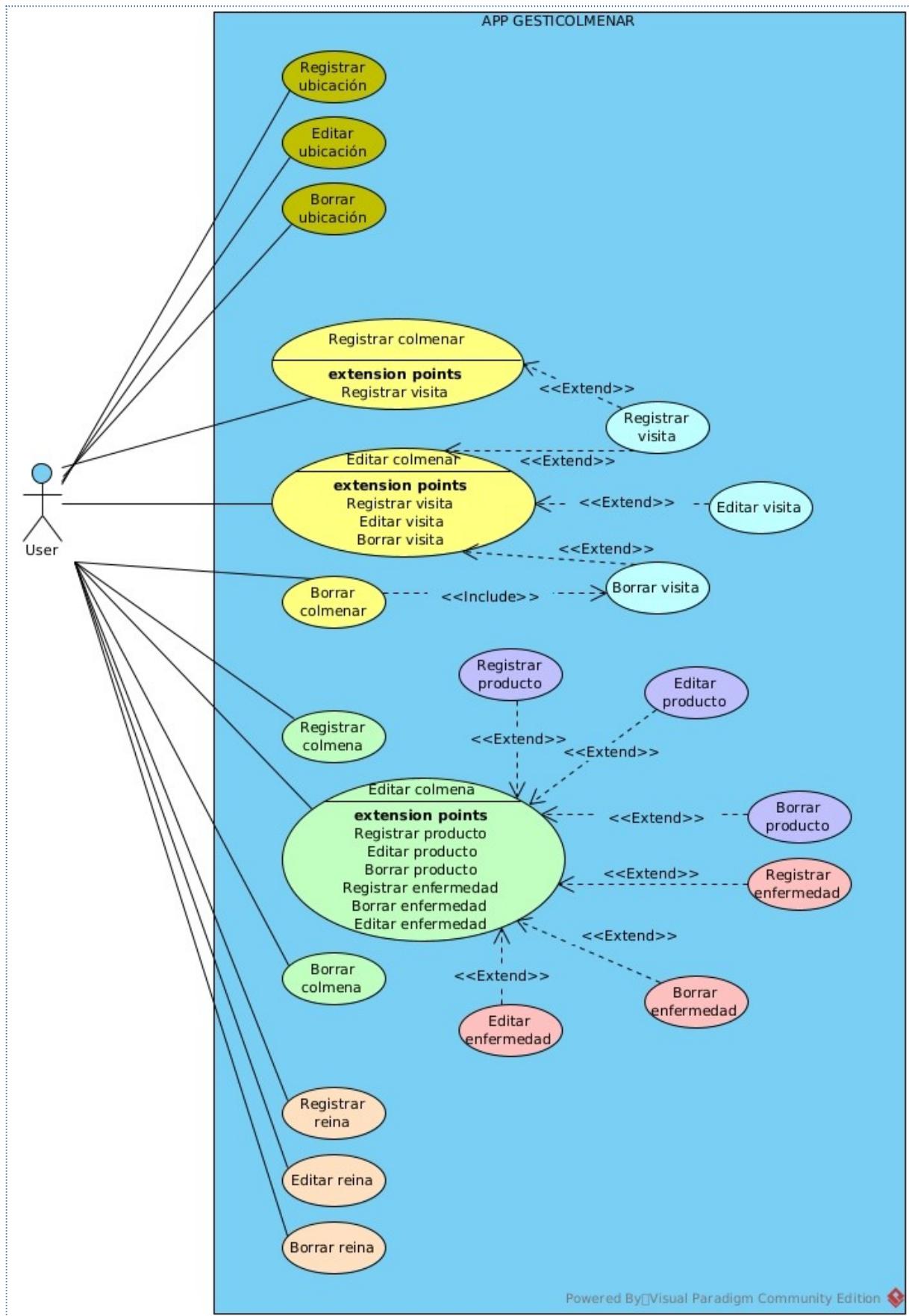
Requisitos:

1. Registro del usuario: la aplicación permite que los usuarios se registren y creen una cuenta.
2. Alta de ubicaciones: se deberán registrar las ubicaciones donde irán cada uno de los colmenares de forma previa al alta de un colmenar, indicando información relevante de dicha ubicación.
3. Alta de colmenares: los usuarios deben tener la capacidad de dar de alta sus propios colmenares, especificando la ubicación de la misma.
4. Alta de abeja reina: el usuario tiene que dar de alta una abeja reina en el sistema de forma previa al alta de una colmena, indicando, datos relevantes de la misma.
5. Alta de colmenas: la aplicación debe permitir al usuario poder dar de alta las colmenas en colmenares previamente dados de alta, especificando además, la reina que gobernará dicha colmena.
6. Indicar enfermedades: el apicultor podrá indicar si alguna colmena presenta alguna o algunas enfermedades o problemas sanitarios, especificando el tipo de las existentes, día en el que se observó el problema y próximo tratamiento en caso de requerirse. Dicho registro de enfermedad, si se indica una fecha de próximo tratamiento y esta es posterior al día actual o el día actual, mostrará un aviso en la sección de tareas. Para quitar el aviso, editaremos la fecha de próximo tratamiento, la eliminaremos o simplemente la dejaremos pasar.
7. Tareas pendientes y próximas visitas: en la gestión de un colmenar, se podrá editar la información del mismo para añadir el día de una visita y

el día de la próxima visita, indicar tareas a realizar y campo de texto para anotaciones. La fecha de visita, si es posterior o el día actual, generará un aviso de tareas pendientes. Para quitar el aviso, editaremos la fecha de próxima visita, la eliminaremos o simplemente la dejaremos pasar.

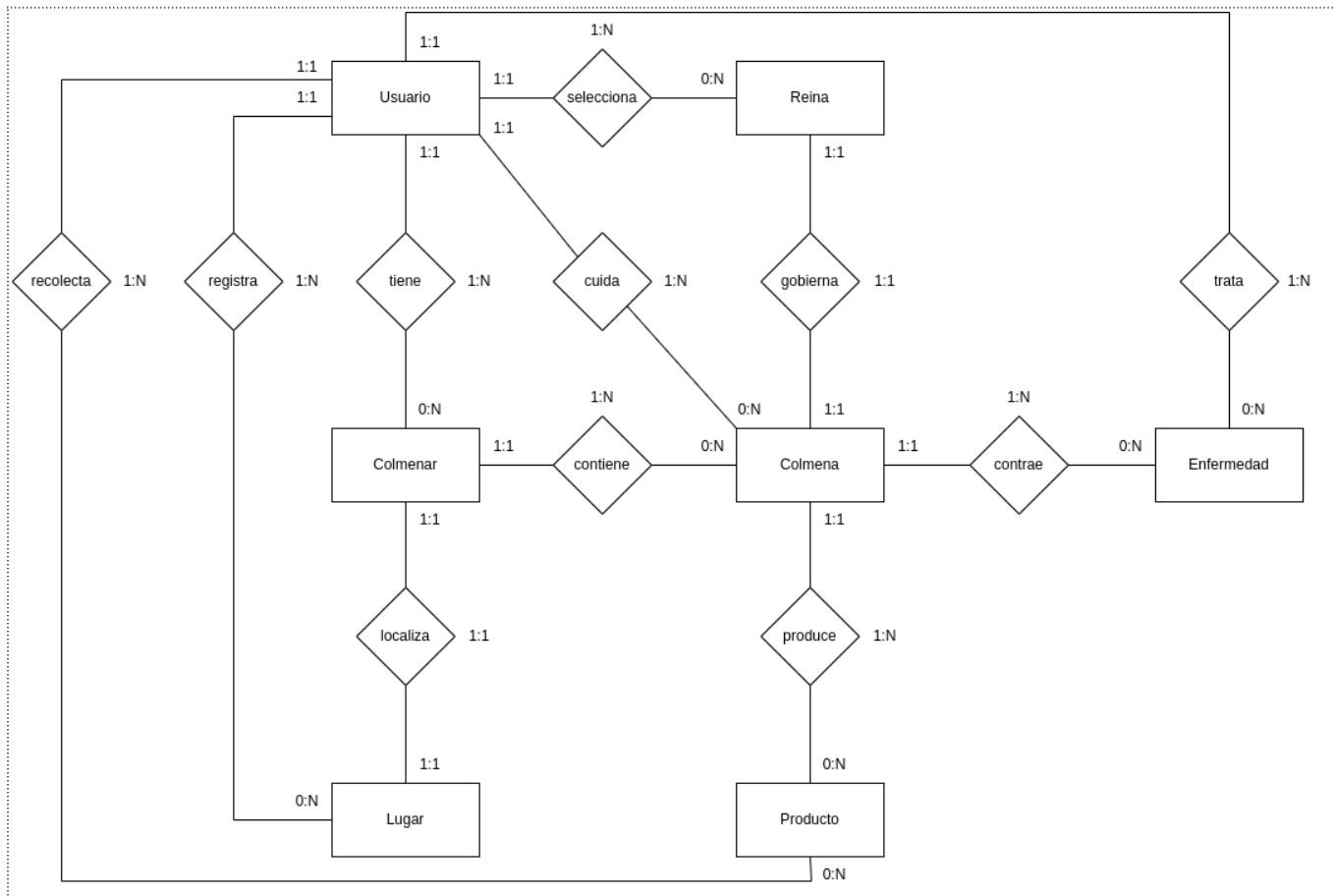
8. Registro de recolección: en cada colmena se podrá editar la información con respecto a la cantidad de kg o gramos recolectados de cada producto a lo largo de la temporada.
9. Gráficas: la aplicación debe mostrar gráficas con los datos de kg totales de miel, polen y gramos de apitoxina, para que los usuarios puedan ver fácilmente el progreso de sus colmenares o la evolución de un colmenar a lo largo de los años si existen datos suficientes para ello.

Caso de uso general



Powered By Visual Paradigm Community Edition

DIAGRAMA ENTIDAD-RELACIÓN



PASO A TABLAS

Users

users (`id`, name, surname, dni, explotation_code, email, email_verified_at, password)

PK: id

| Users | |
|--------------------------------|----------------------|
| <code>id</code> | Unsigned Big Integer |
| <code>explotation_code</code> | String |
| <code>name</code> | String |
| <code>surname</code> | String |
| <code>dni</code> | String |
| <code>email</code> | String |
| <code>email_verified_at</code> | String (nullable) |
| <code>password</code> | String |

id: Campo usado como PK creado por Laravel.

explotation_code: Todo ganadero de abejas ha de estar dado de alta en el REGA y debe tener el código de explotación, el cual ha de ir impreso de forma indeleble en todas y cada una de las colmenas.

name: Nombre del usuario.

surname: Los apellidos del usuario.

dni: Documento identificativo del usuario.

email: Email de contacto del usuario.

Places

places (id, name, catastral_code, poligon, parcel, postal_code, has_water, user_id)

PK: id

FK: user_id → users

| Places | |
|----------------|----------------------|
| id | Unsigned Big Integer |
| name | String |
| catastral_code | String (unique) |
| poligon | String |
| parcel | String |
| postal_code | String |
| has_water | Boolean |
| user_id | Unsigned Big Integer |

id: Campo usado como PK creado por Laravel.

catastral_code: Referencia Catastral del lugar de 20 caracteres alfanuméricos.

poligon: Referencia del polígono cartográfico de la ubicación.

parcel: Referencia de la parcela asociada al polígono.

postal_code: Código postal del lugar donde está el colmenar.

has_water: Booleano que nos indique si el lugar posee acceso a agua.

user_id: Foreign Key

Apiaries

apiaries (`id`, `last_visit`, `next_visit`, `beehives_quantity`, `clear_apiary`, `refill_water`, `collect_honey`, `collect_pollen`, `collect_apitoxine`, `food`, `others`, `user_id`, `place_id`)

PK: `id`

FK: `user_id` → `users`

FK: `places_id` → `places`

UK: `place_id` → `places`

| Apiaries | |
|--------------------------------|----------------------|
| <code>id</code> | Unsigned Big Integer |
| <code>last_visit</code> | Date (nulleable) |
| <code>next_visit</code> | Date (nulleable) |
| <code>beehives_quantity</code> | Integer |
| <code>clear_apiary</code> | Boolean |
| <code>refill_water</code> | Boolean |
| <code>collect_honey</code> | Boolean |
| <code>collect_pollen</code> | Boolean |
| <code>collect_apitoxine</code> | Boolean |
| <code>food</code> | Boolean |
| <code>others</code> | Text (nulleable) |
| <code>user_id</code> | Unsigned Big Integer |
| <code>place_id</code> | Unsigned Big Integer |

id: Campo usado como PK creado por Laravel.

beehives_quantity: Número total de colmenas que contiene el colmenar.

clear_apiary: Necesidad o no de hacer limpiezas en el apiario, desde cortar hierba, limpiar caminos, recoger restos... etc.

refill_water: Indicar si el apiario necesita rellenar el/los depósitos o medios por los que se suministre agua al ganado.

collect_honey/collect_pollen/collect_apitoxine: indicar si hay que ir al apiario a recoger algún producto.

food: Indicar si hay que llevar comida al ganado, normalmente apipasta o asimilados, pues, en la recolección de los productos de la colmena, tanto miel como polen, son, los alimentos que ellas recolectan y por ende, hay que sustituir su alimento por otros, tales como azúcares/edulcorantes/fructosas...

others: Campo de texto para poder realizar anotaciones o añadir recordatorios no contemplados.

user_id: Foreign Key

place_id: Foreign Key

Queens

queens (**id**, race, color, start_date, end_date, is_inseminated, is_zanganera, is_new_blood, **user_id**)

PK: id

FK: user_id → users

| Queens | |
|-----------------------|----------------------|
| id | Unsigned Big Integer |
| race | String |
| color | String |
| start_date | String |
| end_date | String |
| is_inseminated | Boolean |
| is_zanganera | Boolean |
| is_new_blood | Boolean |
| user_id | Unsigned Big Integer |

id: Campo usado como PK creado por Laravel.

race: Las reinas son las que van a trasladar la genética y raza a toda la población de abejas y es un parámetro muy importante para saber si van a ser abejas más o menos agresivas, más o menos productoras de miel o de otros productos de la colmena.

color: Existe un código internacional de colores aceptado por todos los apicultores que identifica el año de nacimiento de una reina: Azul (años terminados en 0 ó 5), Blanco (terminados en 1 ó 6), Amarillo (terminados en 2 ó 7), Rojo (terminados en 3 ó 8) y Verde (terminados en 4 ó 9).

start_date: Año en el que esa reina entra a gobernar una colmena.

end_date: Las reinas tienen un promedio de correcta ovulación fértil de aproximadamente 5 años, normalmente se sustituyen a lo largo del quinto año de su ciclo de vida, por tanto, aquí indicaremos el año en que teóricamente esa reina debe ser sustituida.

is_inseminated: Una reina, ha de realizar el “vuelo nupcial” en el que, uno o diversos zánganos la inseminarán, por tanto, si no es correctamente inseminada pondrá huevos no fértiles, indicándonos si ha sido o no inseminada.

is_zanganera: Una reina, por muchas y diversas razones puede volverse “zanganera”, normalmente, la causa principal es debido a que su “espermanteca” ha llegado a su fin y, empiezan a poner huevos que en su

mayoría darán larvas de zángano y no de obreras, por ende, si se sospecha de esto, hay que indicarlo y observarla, de ser zanganera, habrá que indicarlo y proceder al cambio.

is_new_blood: En la actualidad es muy común adquirir reinas producidas y en muchos casos ya fertilizadas en laboratorios o granjas específicas. Saber si una reina es de nuestras colmenas o es sangre nueva, es un dato importante para saber si la nueva línea es o no interesante.

user_id: Foreign key

Beehives

beehives (id, user_code, type, honey_frames, pollen_frames, brood_frames, user_id, apiary_id, queen_id)

PK: id

FK: user_id → user

FK: apiary_id → apiary

FK: queen_id → queen

| Beehives | |
|---------------|----------------------|
| id | Unsigned Big Integer |
| type | String |
| honey_frames | Integer |
| pollen_frames | Integer |
| brood_frames | Integer |
| user_id | Unsigned Big Integer |
| apiary_id | Unsigned Big Integer |
| queen_id | Unsigned Big Integer |

id: Campo usado como PK creado por Laravel.

type: Existen diferentes modelos de colmenas: Langstroth, Dadant o Layens.

honey_frames: De los marcos/cuadros que hay en la colmena, número de ellos que se destinan a producción de Miel.

pollen_frames: De los marcos/cuadros que hay en la colmena, número de ellos que se destinan a dejarlos con polen para la propia alimentación de las abejas.

brood_frames: De los marcos/cuadros que hay en la colmena, número de ellos que se destinan a la cría de nuevas obreras.

user_id: Foreign key

apiary_id: Foreign key

queen_id: Foreign key

Products

products (id, type, grams, year, user_id, beehive_id)

PK: id

FK: user_id → users

FK: beehive_id → beehives

| Products | |
|------------|----------------------|
| id | Unsigned Big Integer |
| type | String |
| grams | Integer |
| year | String |
| user_id | Unsigned Big Integer |
| beehive_id | Unsigned Big Integer |

id: Campo usado como PK creado por Laravel.

type: los distintos tipos de producto que podríamos sacar de una colmena serían: miel, polen y apitoxina. Existen más productos que se pueden obtener, pero, por norma general no son apiarios al uso, son instalaciones más específicas para la obtención de jalea real y propóleo, del mismo modo, la cera de abeja, no suele comercializarse ya que es empleada para formar los nuevos panales para nuevas campañas.

grams: cantidad del producto extraído, se expresa en gramos ya que, tanto la apitoxina se extrae en pequeñas cantidades, es de fácil conversión a kg para la miel o el polen.

user_id: Foreign key

Diseases

diseases (id, name, treatment_start_date, treatment_repeat_date, user_id, beehive_id)

PK: id

FK: user_id → users

FK: beehive_id → beehives

| Diseases | |
|-----------------------|----------------------|
| id | Unsigned Big Integer |
| name | String |
| treatment_start_date | Date |
| treatment_repeat_date | Date |
| user_id | Unsigned Big Integer |
| beehive_id | Unsigned Big Integer |

id: Campo usado como PK creado por Laravel.

name: Existen distintos tipos de enfermedades relacionadas con las abejas, algunas de ellas parasitarias, tales como Varroa Destructor, Loque Americana... etc.

treatment_start_date: fecha en la que se lleva a cabo el tratamiento de la colmena.

treatment_repeat_date: fecha en la que se debería volver a repetir el tratamiento, en caso necesario.

user_id: Foreign key

beehive_id: Foreign key

DIAGRAMA FÍSICO

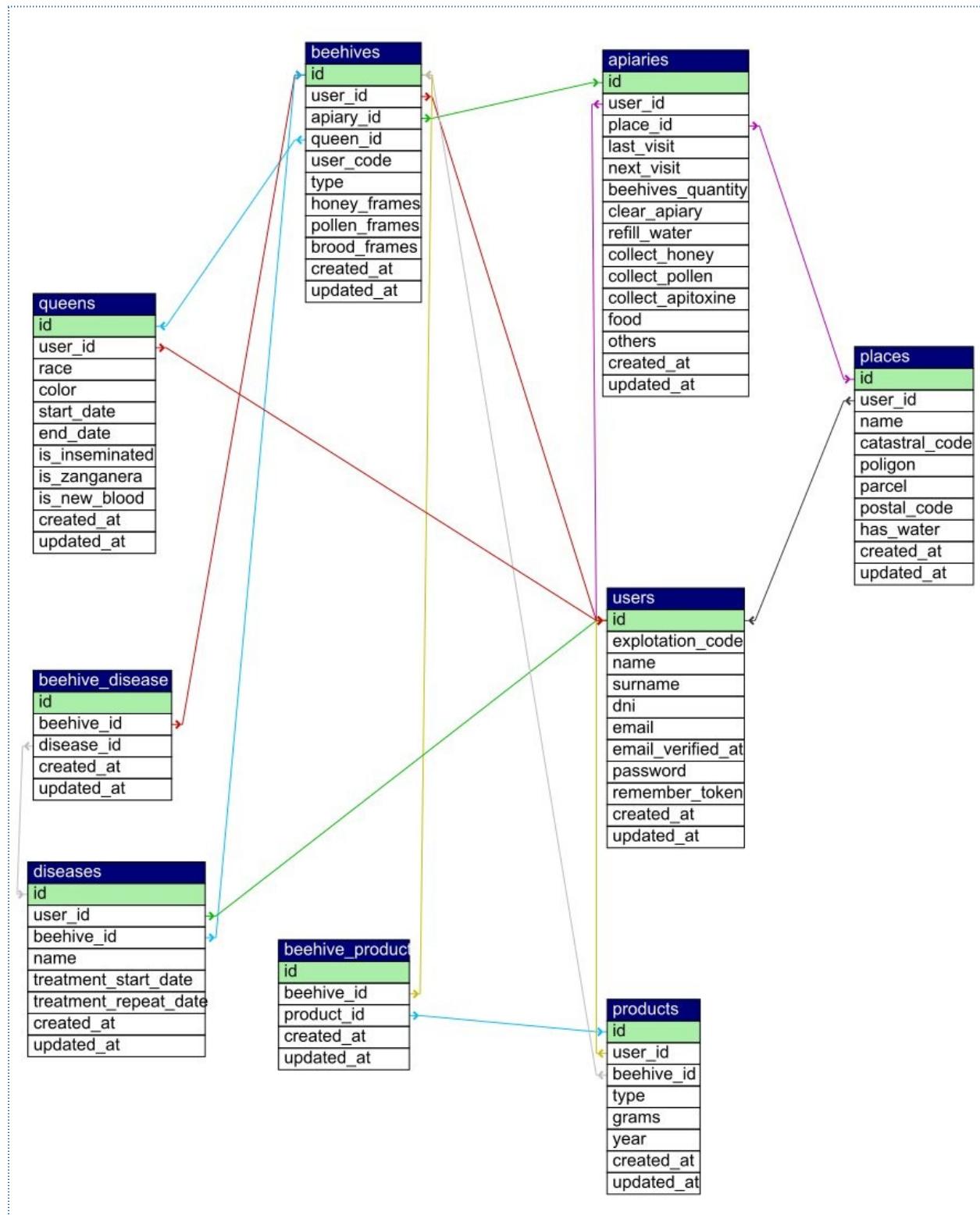
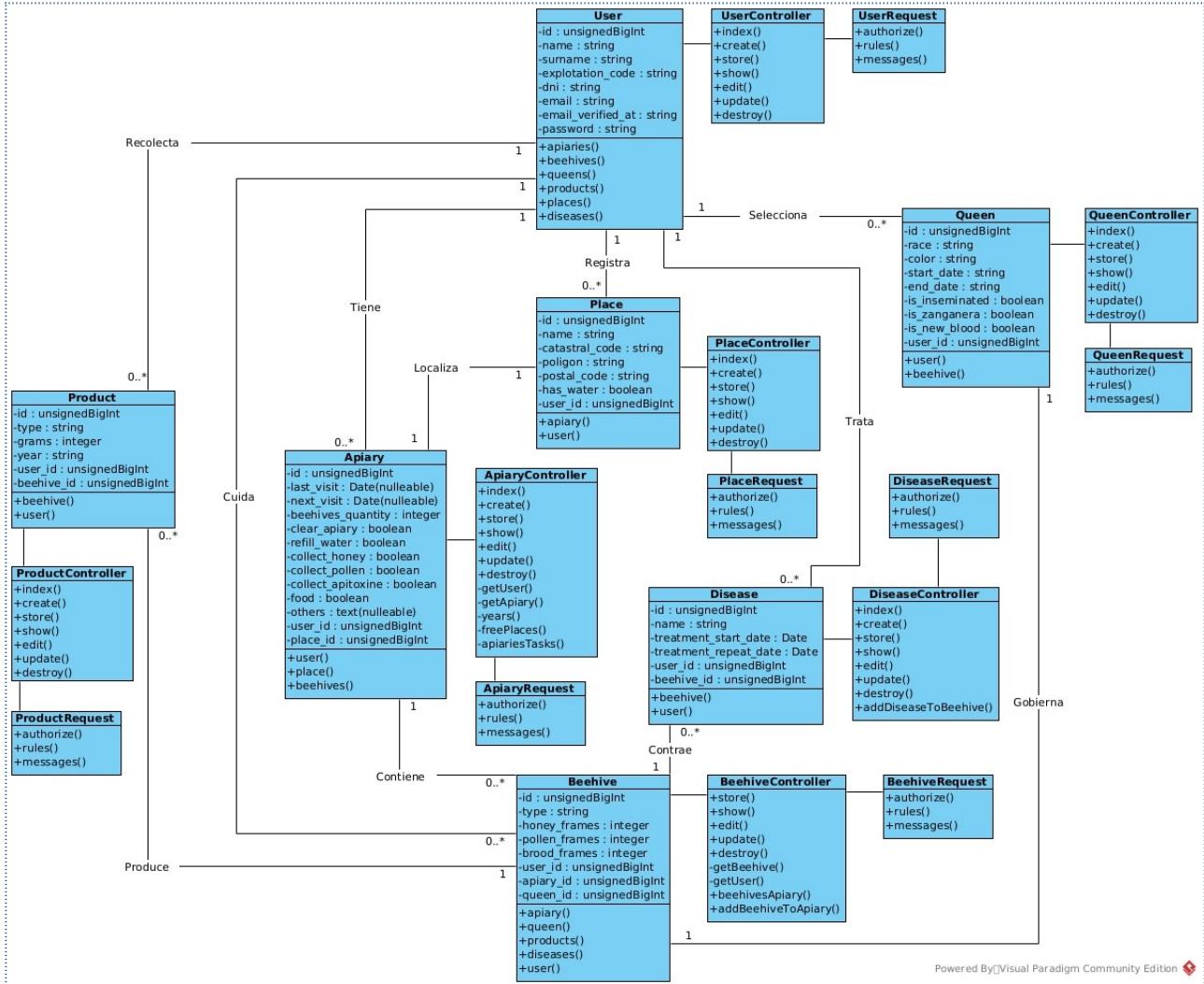


DIAGRAMA DE CLASES



Powered By Visual Paradigm Community Edition

DIAGRAMA DE SECUENCIAS

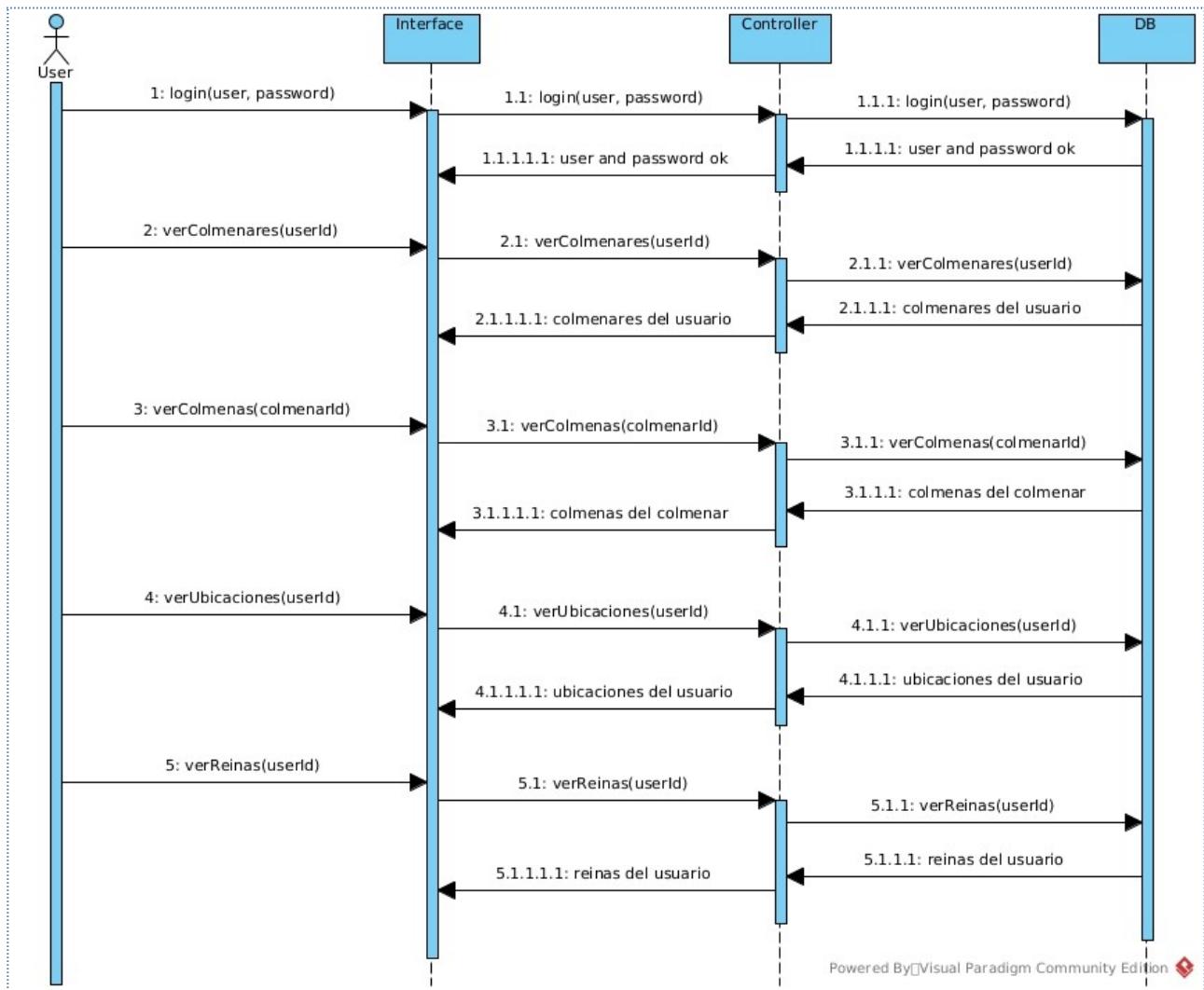
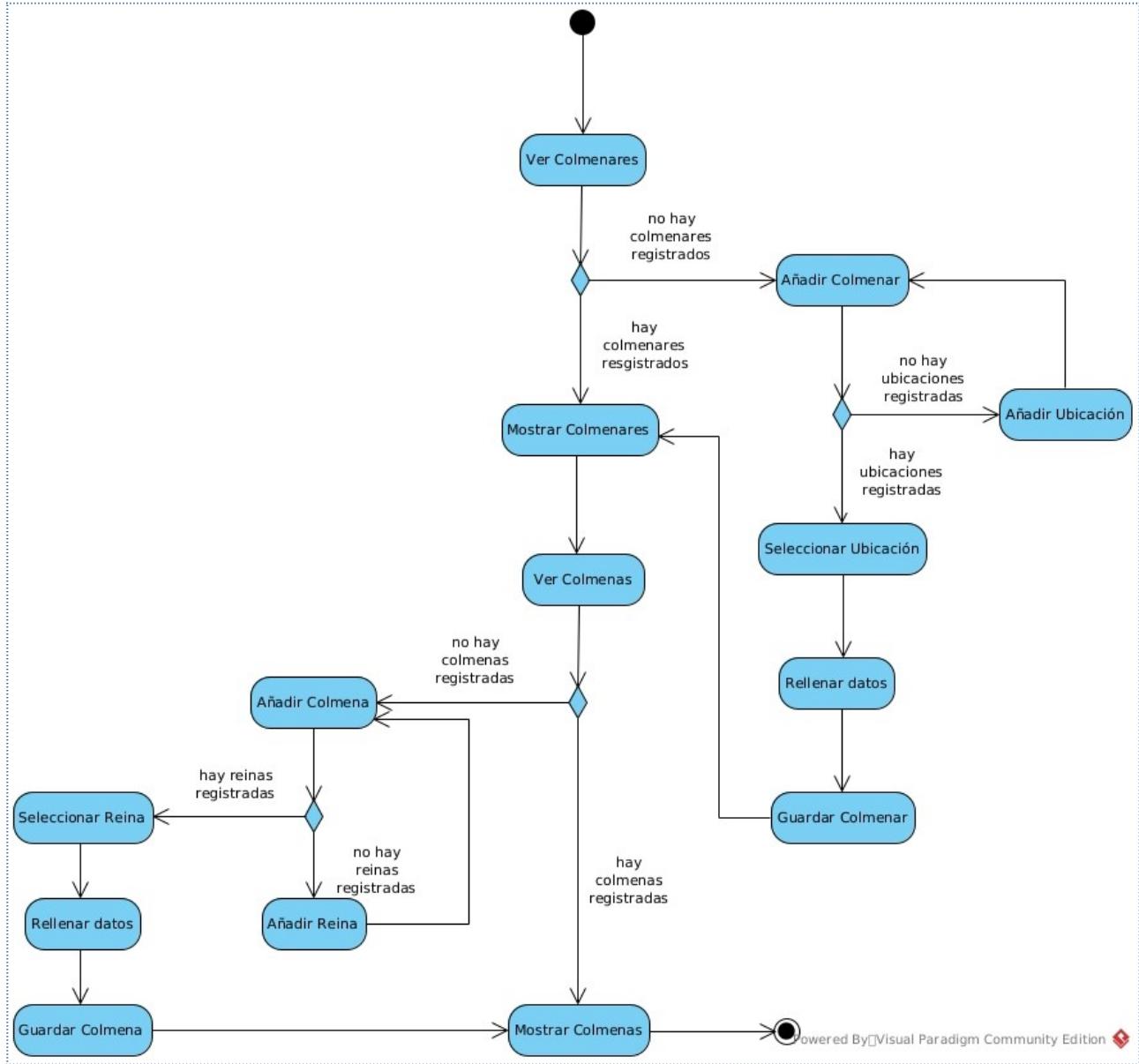
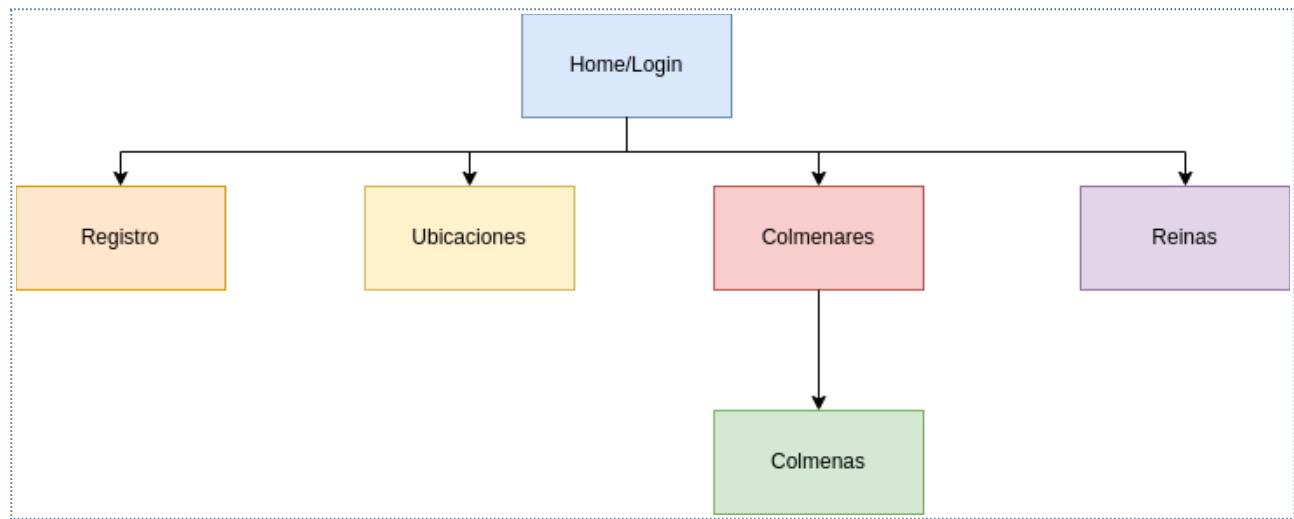


DIAGRAMA DE ACTIVIDADES

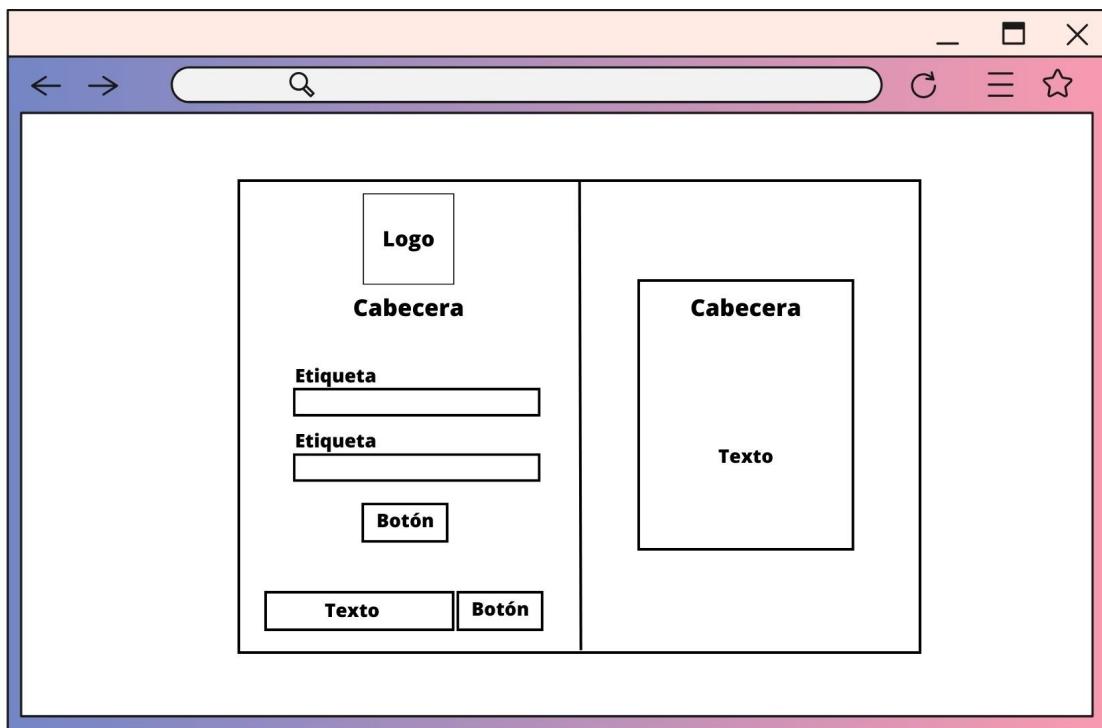


SITE MAP

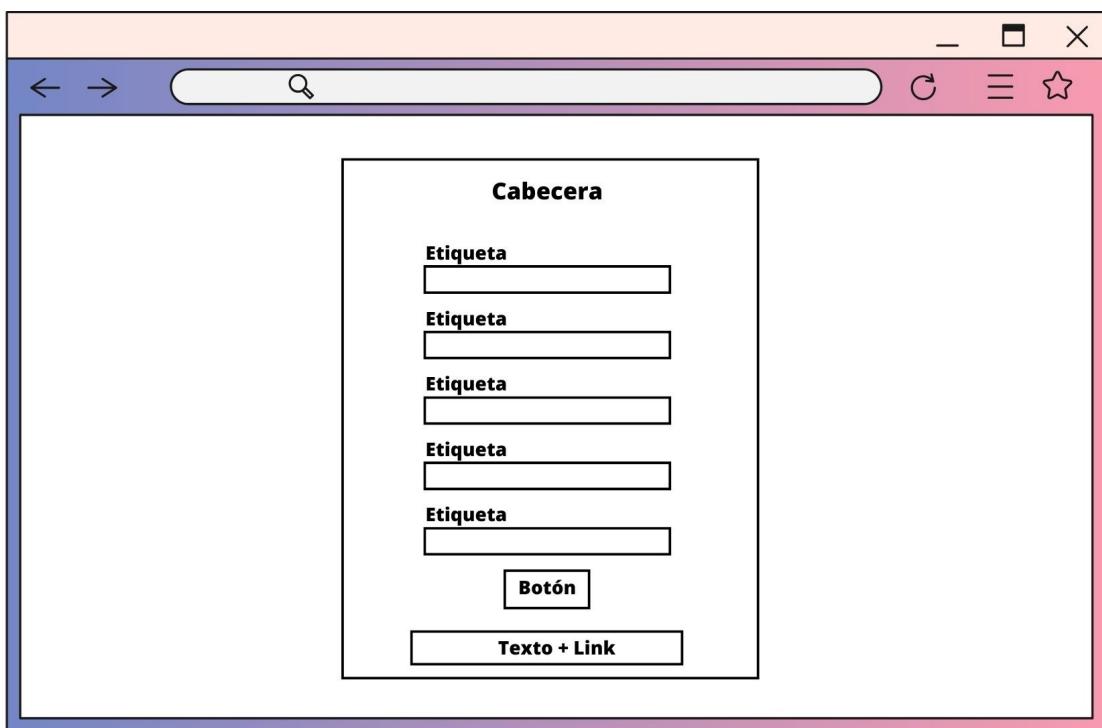


MOCKUPS DESKTOP

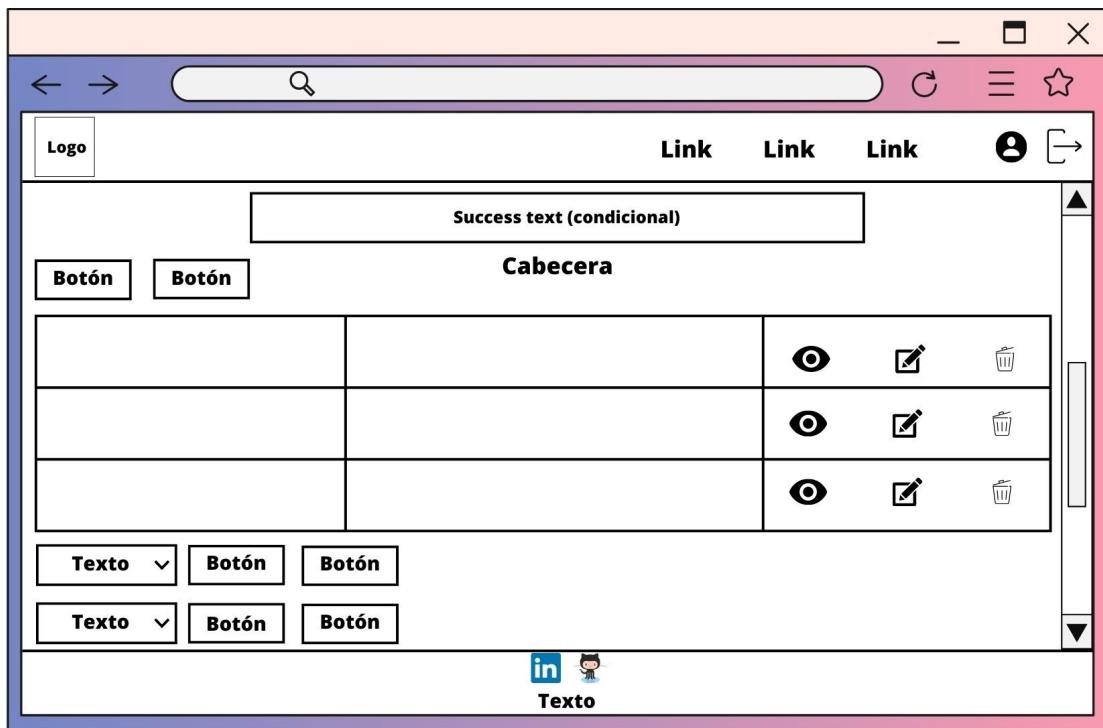
Login



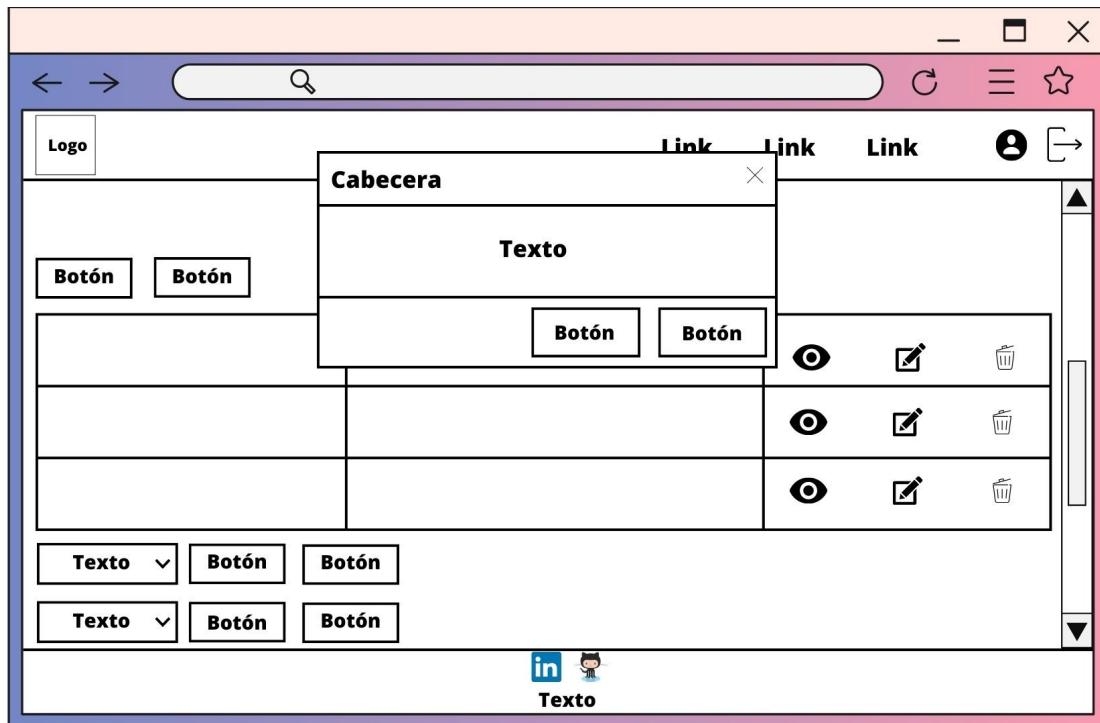
Registro



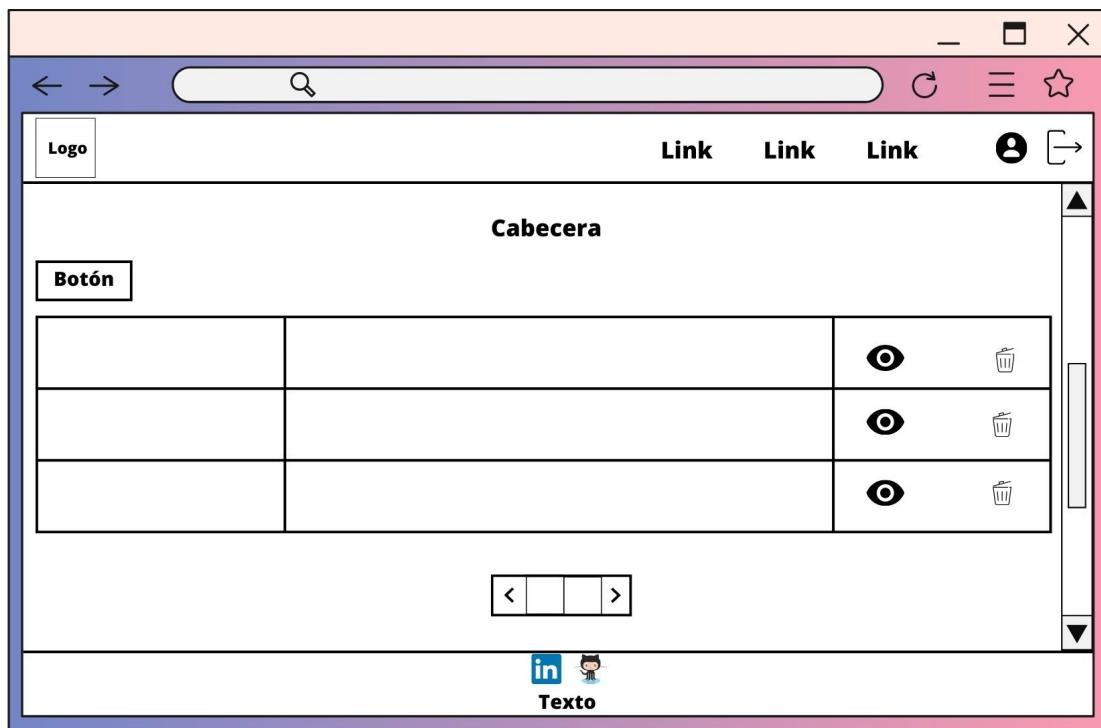
Vista principal/Colmenares



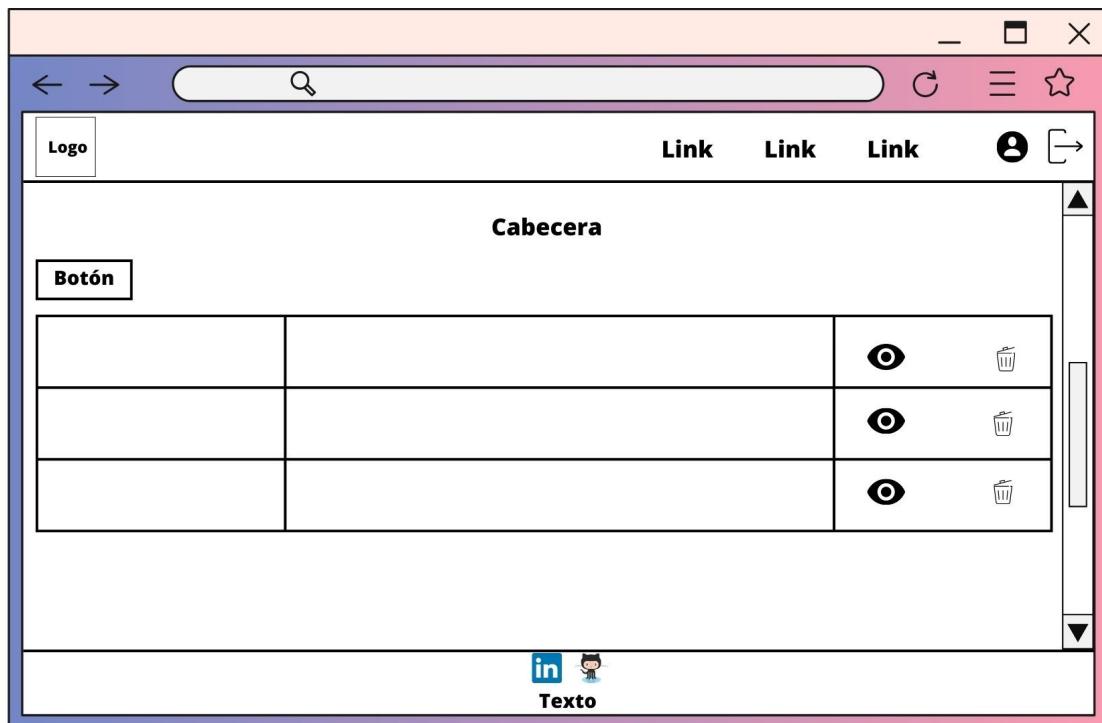
Vista Colmenares Modal Borrar



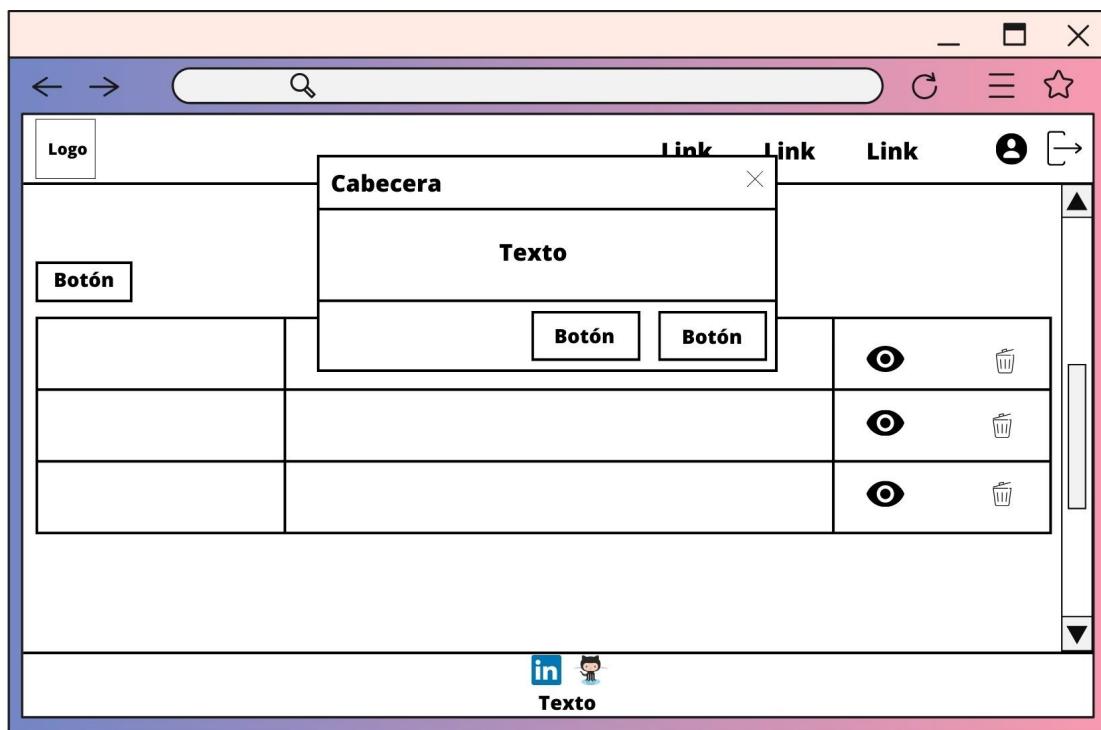
Vista Colmenas



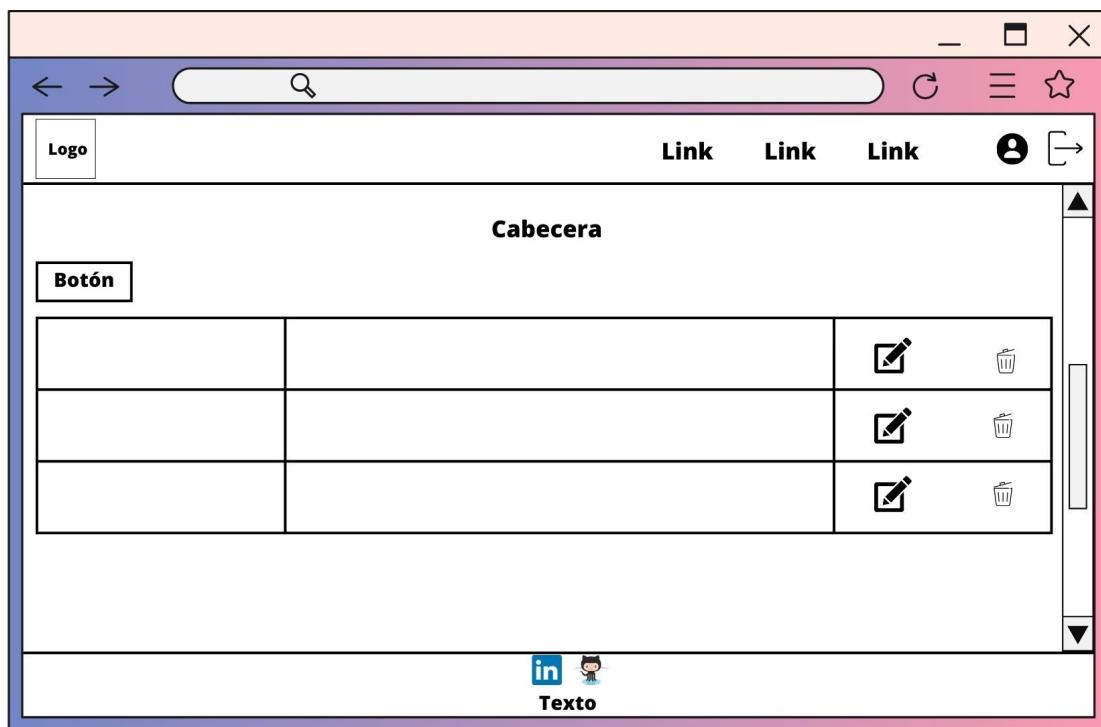
Vista Ubicaciones



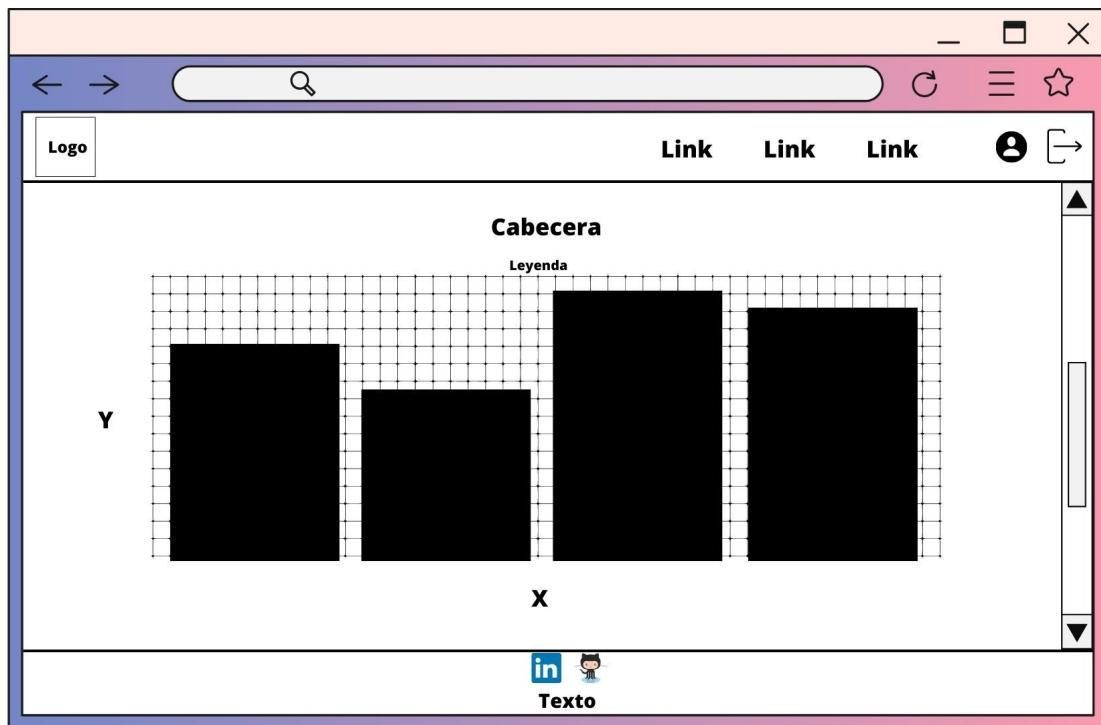
Vista Ubicaciones Modal Borrar



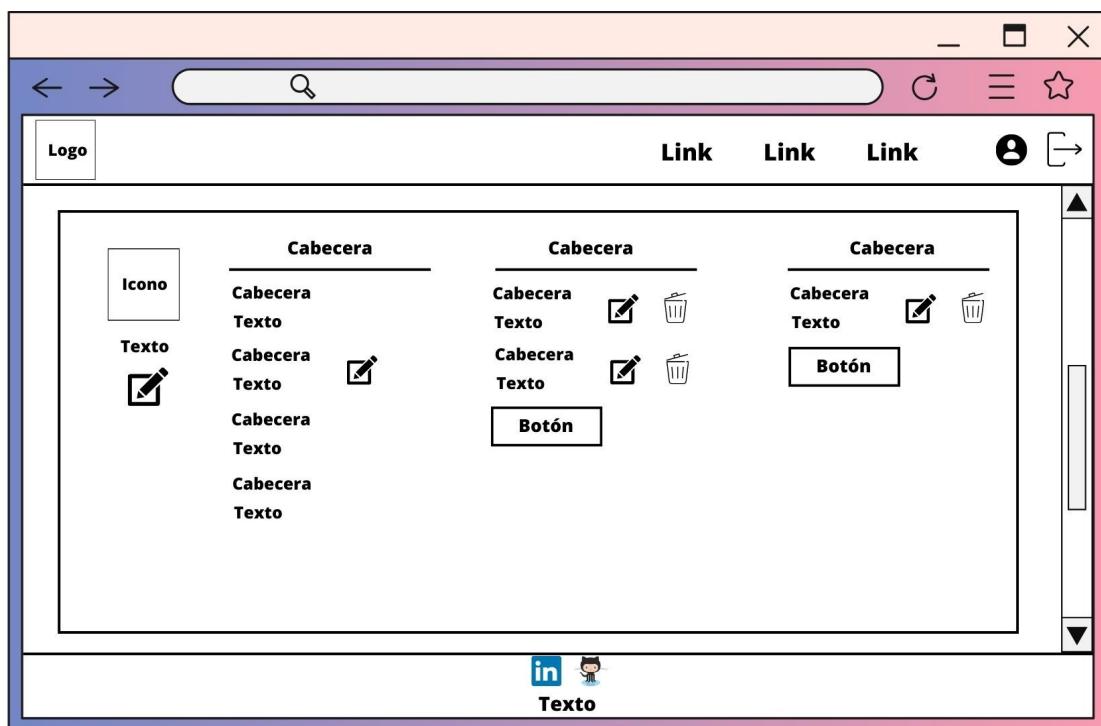
Vista Reinas



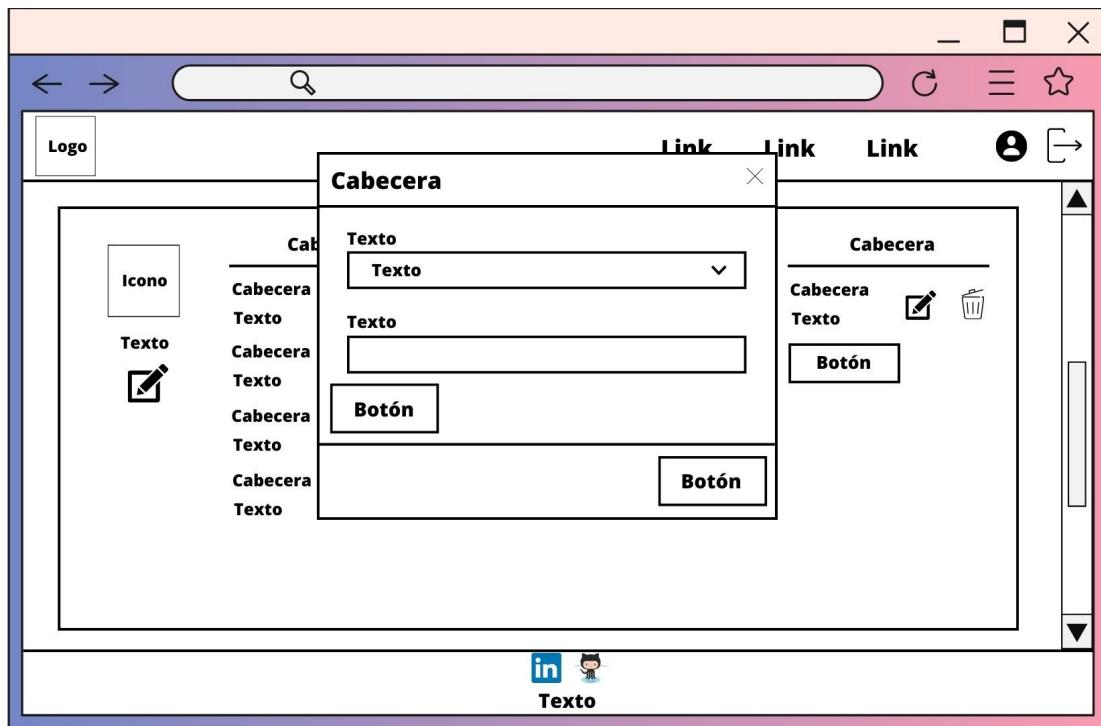
Vista distintas Gráficas



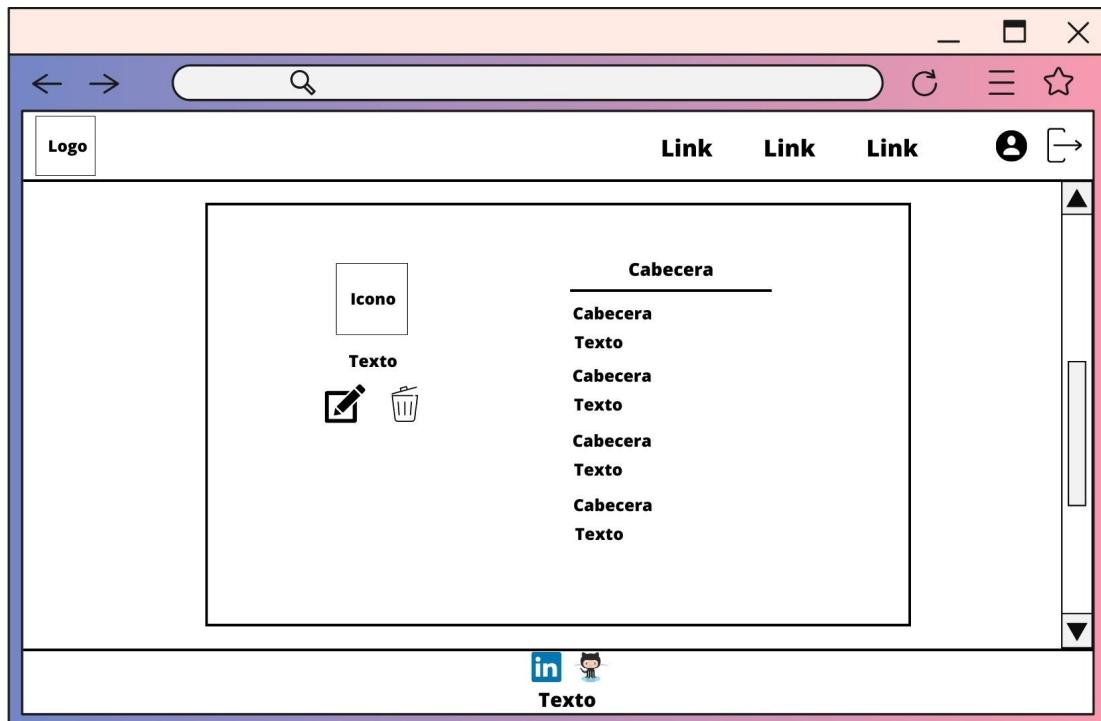
Vista detalles Colmena



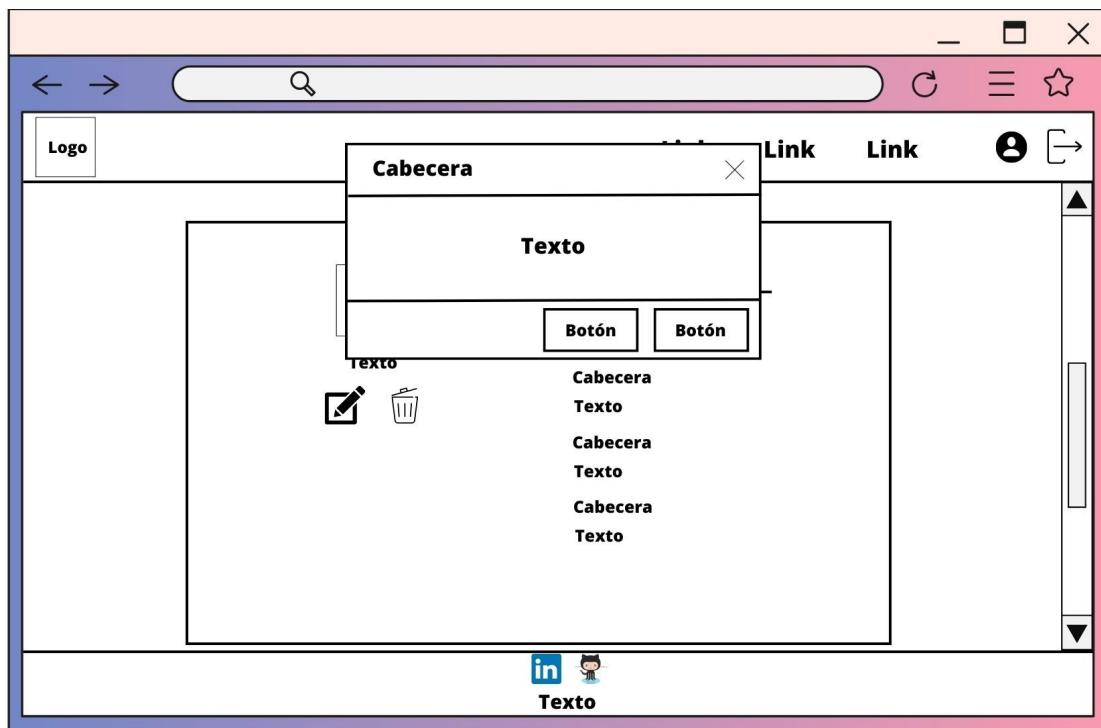
Vista Colmena Modal Añadir Producto



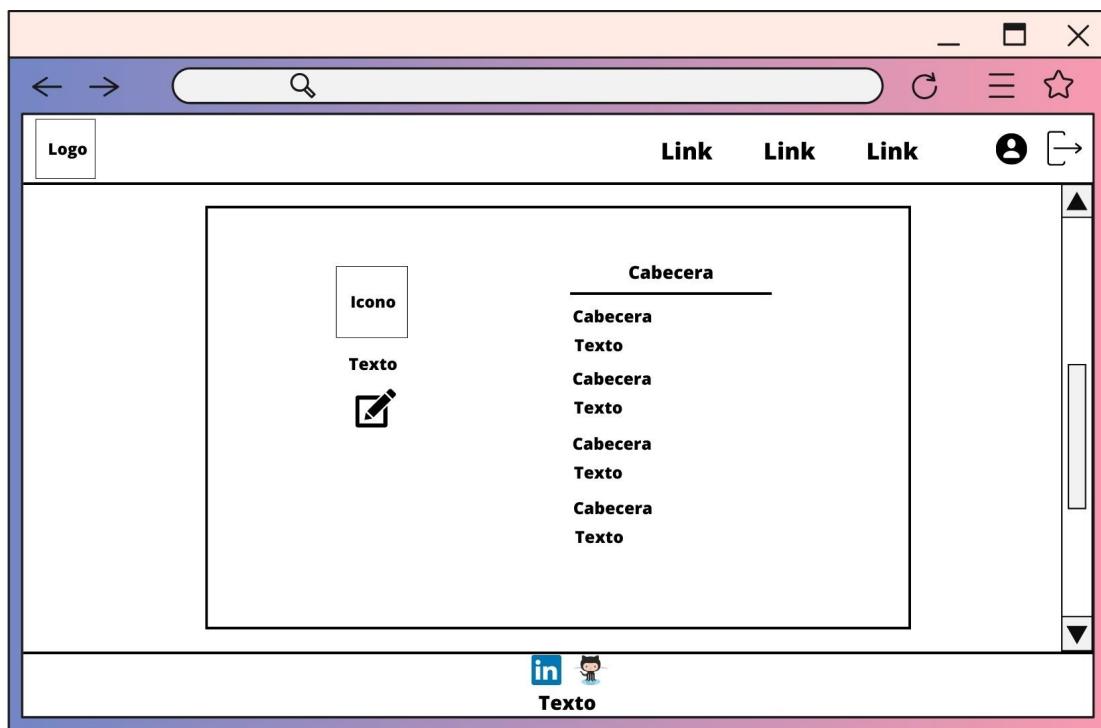
Vista detalles Usuario



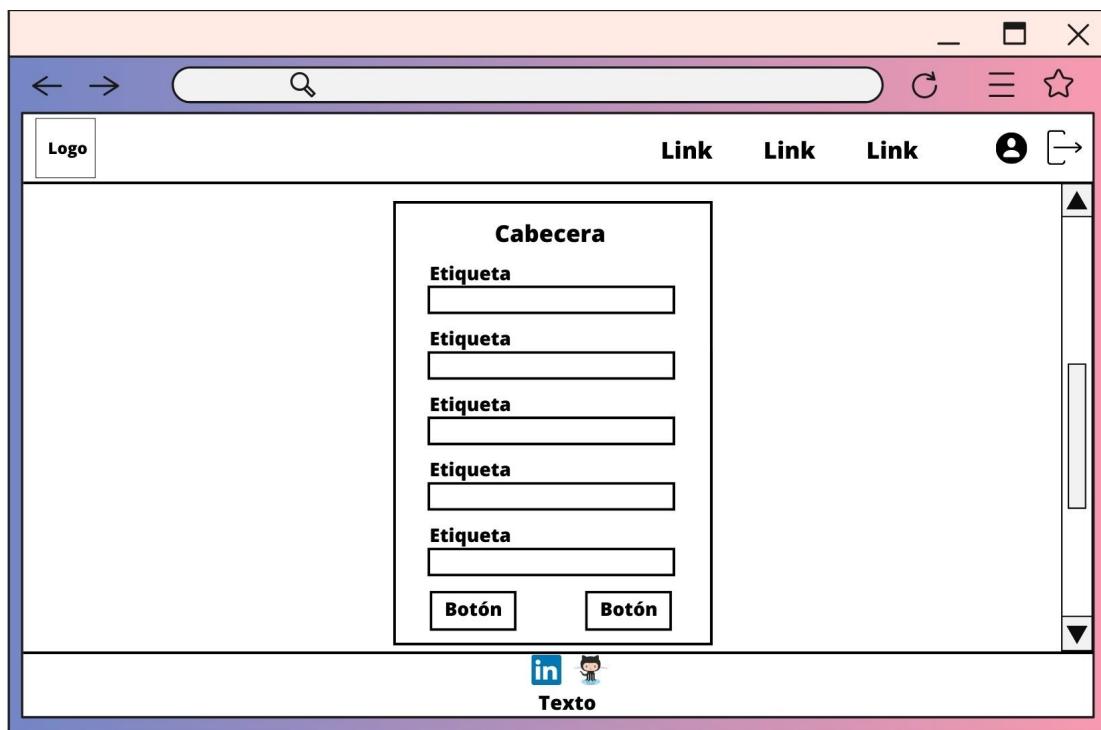
Vista Modal Borrar Usuario



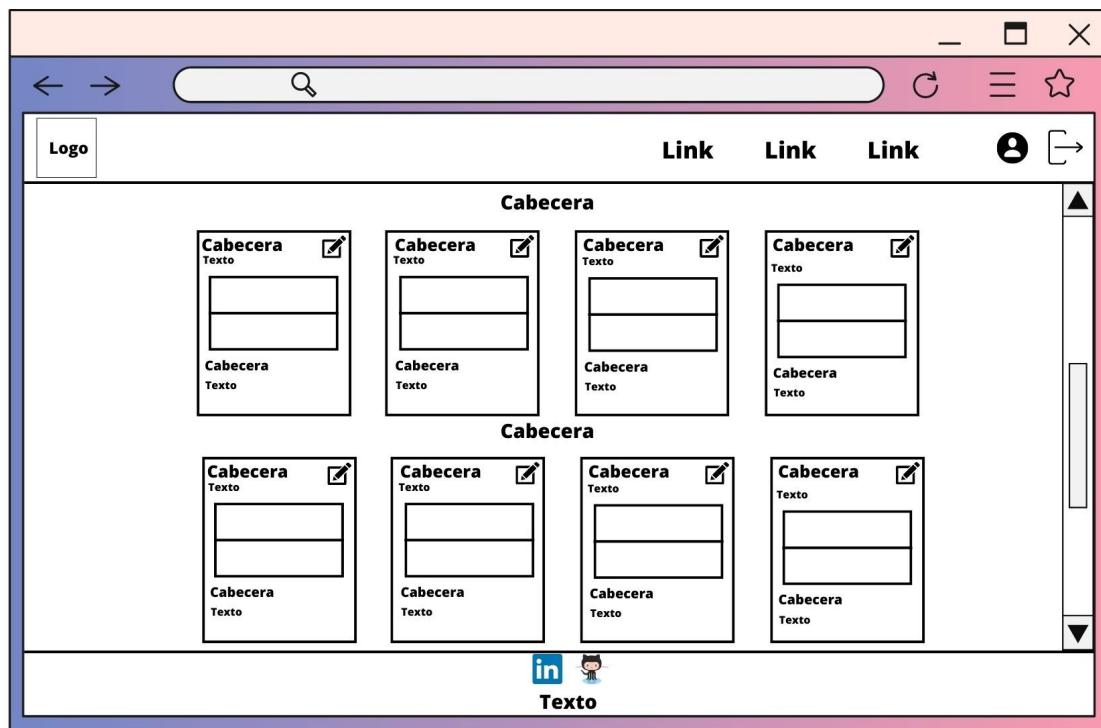
Vista detalles Ubicación



Vista Formularios (edición/creación)

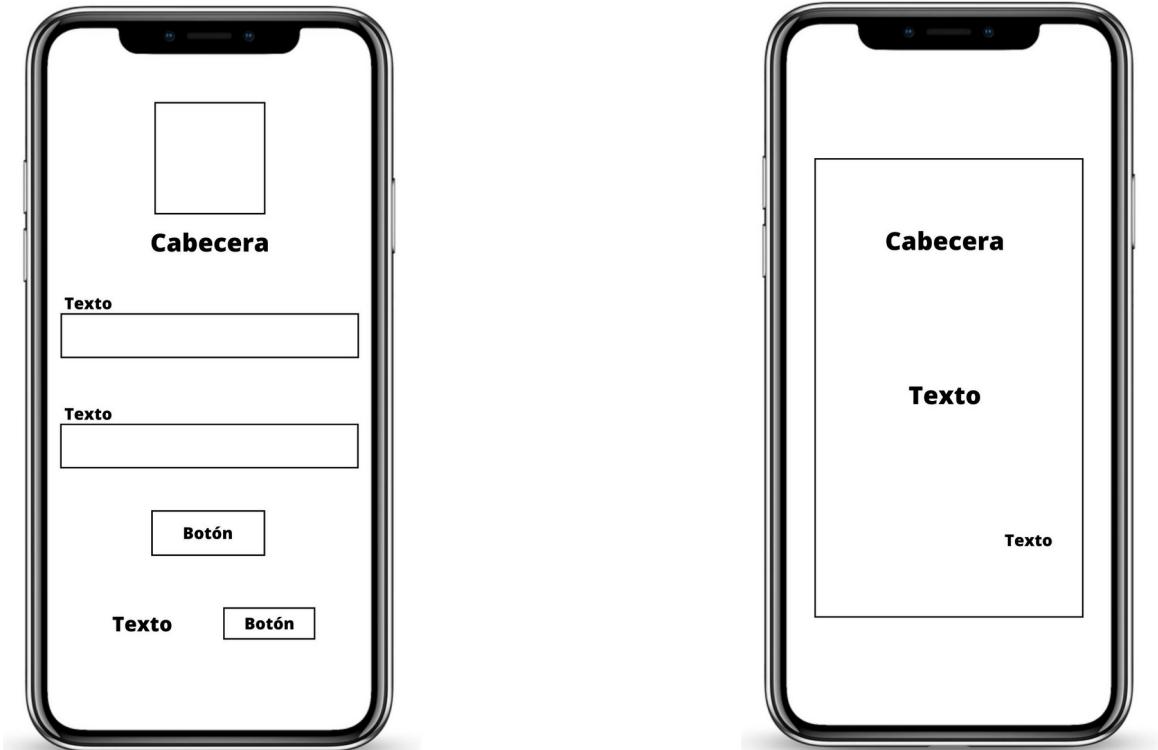


Vista Tareas Pendientes TODO

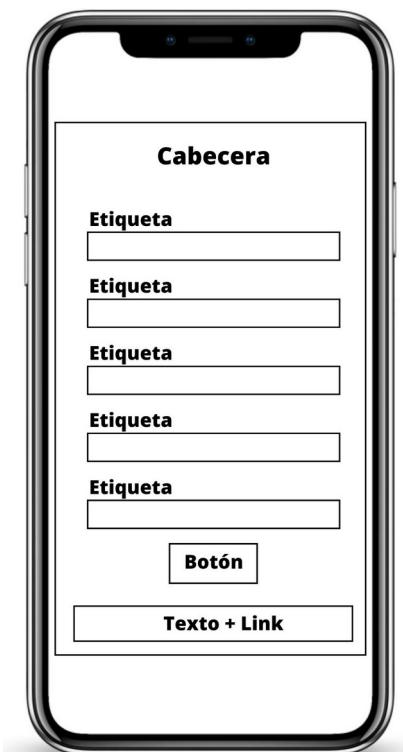


MOCKUPS MÓVIL

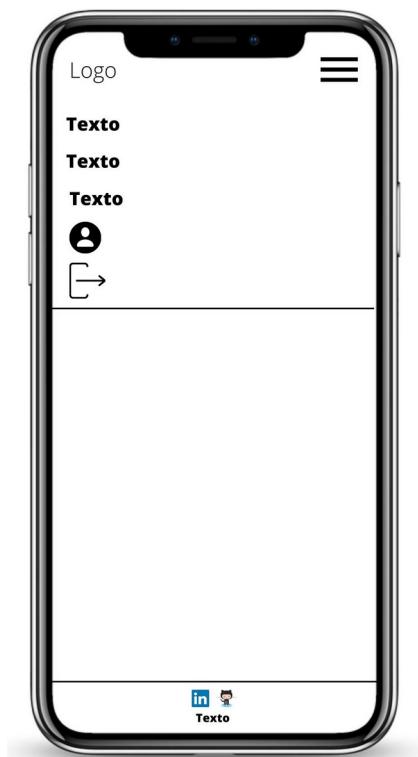
Vista Login



Vista Registro



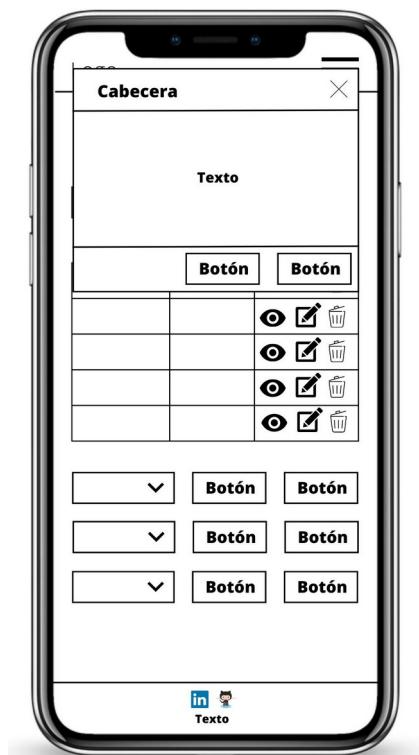
Vista menú desplegado



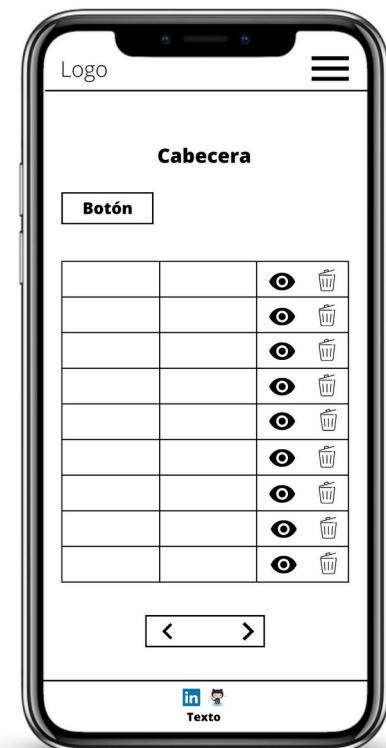
Vista principal/Colmenares



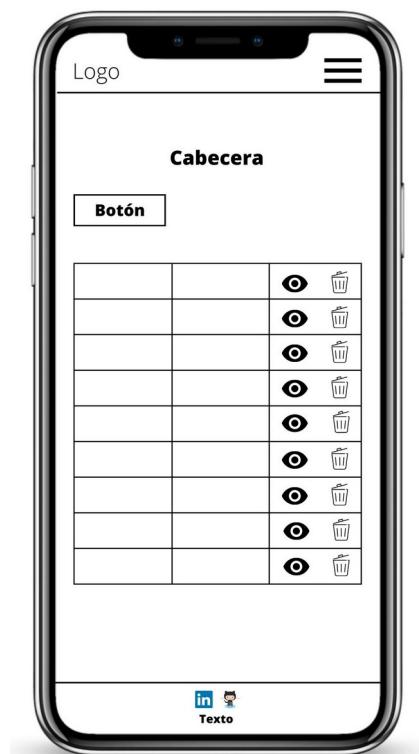
Vista Colmenares Modal Borrar



Vista Colmenas



Vista Ubicaciones



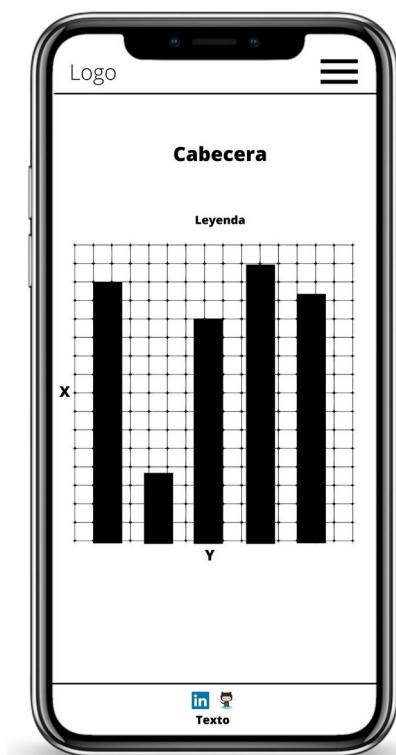
Vista Ubicaciones Modal Borrar



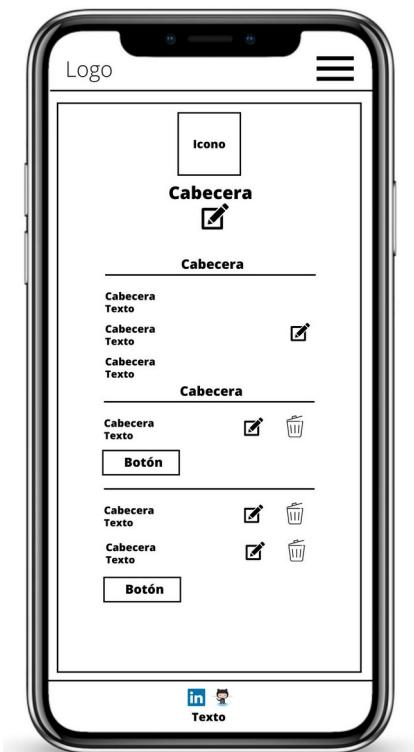
Vista Reinas



Vista distintas Gráficas



Vista detalles Colmena



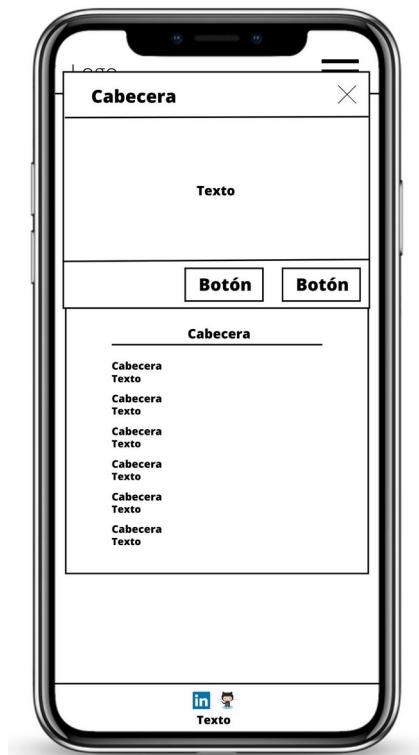
Vista Colmena Modal Añadir Producto



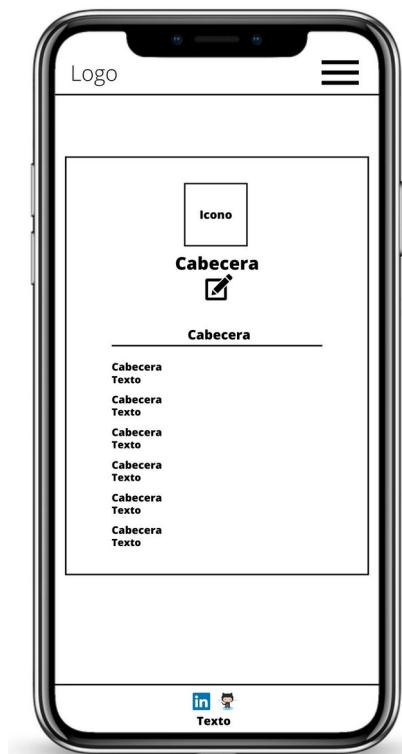
Vista detalles Usuario



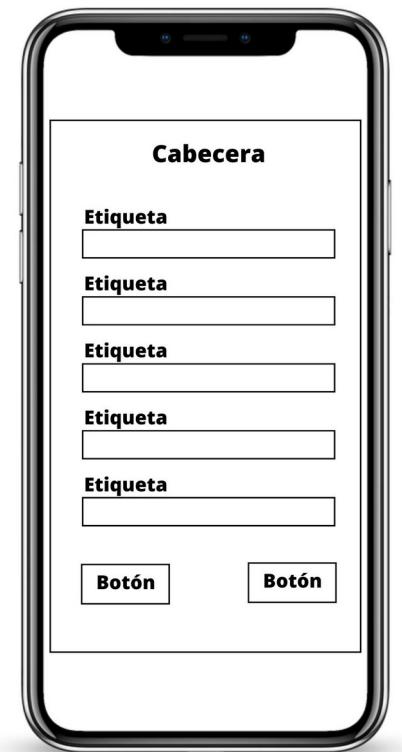
Vista Modal Borrar Usuario



Vista detalles Ubicación



Vista Formularios (edición/creación)



Vista Tareas Pendientes TODO



CODIFICACIÓN

Tecnologías elegidas y su justificación

En este proyecto he elegido como framework de PHP a Laravel, ya que, es muy popular, relativamente fácil de aprender y usar y me permite desarrollar la lógica de la aplicación web de manera estructurada y escalable. Además, Laravel ofrece una gran cantidad de características y herramientas, como el enrutamiento, la gestión de sesiones y autenticación de usuarios, manejo de la base de datos mediante el ORM de Eloquent...

Así mismo, existen multitud de librerías externas y una gran comunidad, por ello, también se ha usado una librería para mostrar los datos en formato gráfico, en este caso, se ha usado la librería “ConsoleTV/Charts”: <https://packagist.org/packages/consoletvs/charts>

Así mismo, se ha usado junto a Laravel, Blade, que es un sistema de plantillas/vistas para el frontend que está integrado con el framework de Laravel, permitiendo de esta manera, separar la lógica de la presentación, manteniendo el código limpio y organizado.

Por otro lado y para visualizar el sitio correctamente y con un estilo limpio, se ha utilizado el framework de Bootstrap, que no deja de ser un conjunto de herramientas para desarrollar sitios y aplicaciones web. Ofrece una gran cantidad de estilos y componentes predefinidos para poder crear sitios responsive y que sea atractivo visualmente.

Para dar algunos estilos que no están incluidos con Bootstrap se ha empleado CSS y no SCSS ya que la cantidad de líneas de código han sido muy pocas no aportando nada el uso de SCSS.

Finalmente, para el sistema de gestión de la base de datos se ha usado uno de los más populares y más ampliamente utilizados, MySQL. Se ha empleado dicho sistema ya que Laravel lo incorpora de forma nativa y se puede integrar de forma sencilla la aplicación con la base de datos.

Laravel es un framework de PHP muy popular y fácil de usar que te permite desarrollar la lógica de la aplicación web de manera estructurada y escalable. Laravel ofrece una gran cantidad de características y herramientas, como el enrutamiento, la gestión de sesiones, la autenticación de usuarios y la base de datos ORM.

En resumen, las tecnologías que elegidas considero que son una muy buena combinación para desarrollar GESTICOLMENAR de manera eficiente y efectiva. Laravel te permite manejar la lógica de la aplicación, Blade te ayuda a separar la lógica de la presentación, Bootstrap te ofrece una amplia gama de componentes para la interfaz de usuario, MySQL te permite almacenar y recuperar datos de manera eficiente y CSS te permitirá personalizar la apariencia de tu aplicación.

Entorno servidor

En cuanto a la descripción general, diría que la aplicación será alojada en un servidor web como Apache o Nginx y base de datos MySQL.

En cuanto al tema de la seguridad, usando Laravel, se disponen de herramientas de validación de datos integradas para los campos de entrada de los formularios, por lo tanto se evitan así los ataques de inyección SQL.

Se usa encriptación para datos sensibles como podría ser la contraseña.

Además, Laravel nos incluye un sistema de autenticación y autorización para restringir el acceso a ciertas partes de la aplicación sólo a usuarios autenticados, protegiendo también el enrutamiento de la misma forma.

Entorno cliente

Se ha asegurado que la aplicación sea compatible con diferentes navegadores tales como Chrome, Firefox y Brave, así mismo, también se ha validado que se visualice correctamente en dispositivos móviles mediante las herramientas de desarrollo que incorpora Chrome y su posibilidad de visualizar en distintos dispositivos móviles, asegurando de este modo que GESTICOLMENAR es una app responsive y que funciona como es debido.

Para obtener un correcto funcionamiento responsive, se han basado las vistas en Bootstrap a través de Blade, proporcionando esa adaptación correcta, independientemente del viewport, posicionando los elementos en las posiciones correctas para su legibilidad, usabilidad y con un buen user experience.

DESPLIEGUE

Descripción de la instalación

Lo primero que deberemos hacer en nuestra aplicación será ir al archivo .env y modificar dos líneas para dejarla preparada para producción.

```
2 APP_ENV=production  
3 APP_DEBUG=false
```

En el mismo archivo .env debemos modificar los datos necesarios para la base de datos, definiendo las variables que coincidan con la configuración de la base de datos de MySQL que configuremos en el hosting (nombre de la base de datos, nombre de usuario, password, puerto... etc.).

Deberemos lanzar ahora una serie de instrucciones necesarias para preparar la aplicación según la documentación oficial de la página web de Laravel (<https://laravel.com/docs/10.x/deployment>):

Con esta instrucción: `composer install --optimize-autoloader -no-dev` nos aseguramos de optimizar el mapa interno que se encarga de cargar las distintas clases de Composer para que pueda encontrar rápidamente el archivo adecuado en cada carga.

Ahora usaremos: `php artisan config:cache` que combinará todas las configuraciones de nuestra aplicación de Laravel en un único archivo en caché, lo que reduce en gran medida la cantidad de llamadas que debe realizar la aplicación para cargar los valores de configuración.

Al igual que el comando anterior, con este: `php artisan route:cache` lo que haremos será cachear todas las rutas de la aplicación durante el proceso de despliegue.

Finalmente, también cachearemos las vistas con: `php artisan view:cache`, este comando nos precompila en caché todas las vistas Blade para que no se tengan que compilar en tiempo de demanda, lo que mejorará el rendimiento en cada solicitud de una vista.

Con todo esto, procederemos a comprimir nuestra aplicación y desde el panel de control de nuestro hosting, subiremos el .zip. Una vez en nuestro hosting, procederemos a descomprimir el archivo y moveremos el contenido de nuestra carpeta public a la carpeta public_html del servidor y borraremos la carpeta

public pues habrá quedado sin contenido. El resto del contenido, deberá quedar dentro de la carpeta que posee nuestro nombre de dominio.

Configuraremos ahora el archivo index.php para que pueda servir nuestra aplicación. Para ello, en la carpeta public_html tendremos nuestro index.php, el cual editaremos y añadiremos el siguiente código:

```
7
8 | $app->bind('path.public', function () {
9 |   return __DIR__;
10| });

11
```

Con todo esto, deberíamos ya, poder ver nuestra aplicación desplegada en nuestro hosting.

El uso de CDNs

Mejora el rendimiento de la página: Al utilizar un CDN, el contenido estático como el CSS y el JavaScript se almacenan en servidores en todo el mundo, lo que permite una entrega más rápida al usuario final. Esto reduce la latencia de la página, mejorando su velocidad y rendimiento.

Ahorra tiempo y recursos: Si bien instalar Bootstrap en Laravel puede ser una opción viable, puede ser un proceso tedioso y consume tiempo y recursos del servidor. Al utilizar un CDN, se ahorra tiempo y recursos, ya que no es necesario descargar e instalar Bootstrap en el servidor.

Actualizaciones automáticas: Con un CDN, las actualizaciones de Bootstrap se aplican automáticamente en todo el sitio web. Esto significa que no hay necesidad de actualizar manualmente cada archivo en el servidor, lo que puede ser un proceso propenso a errores.

Mayor escalabilidad: Un CDN es altamente escalable, lo que significa que puede manejar un gran volumen de tráfico sin afectar la velocidad de la página.

Reducción del riesgo de fallos: Al utilizar un CDN, se reduce el riesgo de fallos del servidor ya que el contenido está distribuido en múltiples servidores. Si un servidor falla, el contenido estático todavía se puede servir desde otros servidores, lo que garantiza la disponibilidad del sitio.

En resumen, utilizar un CDN para servir Bootstrap en lugar de instalarlo en Laravel ofrece beneficios significativos, como mejoras en el rendimiento, ahorro de tiempo y recursos, actualizaciones automáticas, mayor escalabilidad y reducción del riesgo de fallos. Todo esto es lo que me ha hecho decantarme por la opción del CDN vs la instalación de Bootstrap dentro del proyecto.

Despliegue hosting gratuito 000webhost

Se hace búsqueda de hostings que no requieran el uso de tarjetas de crédito ni de pago, que sean completamente libres de uso, que dejen hacer una base de datos relacional y que se pueda instalar una aplicación de Laravel. Tras bastantes horas de búsqueda he dado con 000webhost que me ha permitido hacer el despliegue real de la aplicación. Los pasos no están detallados ni explicados en la web del hosting, pero buscando información adicional he podido hacer dicho despliegue unido a los pasos descritos con anterioridad.

He tenido que descargar un archivo llamado php_unzipper (<https://github.com/ndeet/unzipper/blob/master/unzipper.php>), que será subido al hosting junto con el zip del proyecto tras realizar los pasos comentados en el punto anterior para prepararlo para despliegue.

Desde phpmyadmin en mi caso, he hecho un export de la base de datos, archivo que también será subido al hosting.

Pasos:

1. Abrimos el gestor de archivos del hosting y subimos dentro de la carpeta "public_html" el zip del proyecto, el archivo unzipper.php y nuestro archivo de export de la base de datos.
2. Desde el navegador, pondremos en la barra de navegación nuestro dirección gratuita apuntando al archivo unzipper.php tal que: gesticolmenar.000webhostapp/unzipper.php
3. Tras terminar la descompresión, moveremos todo a la raíz del hosting con el gestor de archivos.
4. Borramos la carpeta public_html y, la carpeta public de nuestro proyecto de Laravel, la renombraremos ahora por public_html.

5. De nuestro proyecto, nos iremos a la carpeta app/providers y editaremos el archivo AppServiceProvider.php y buscaremos la función “register()” y la dejaremos como sigue (en cada edición de archivo, siempre acordarse de darle a save&exit):

```
public function register() {  
  
    $this→app→bind('path.public', function()  
  
    {  
  
        return base_path('public_html');  
  
    });  
  
}
```

6. Abriremos ahora el archivo .env que se encuentra en la raiz del proyecto y copiaremos la app_key (sin copiar el texto base64 que precede a la key).
7. Nos desplazamos a la carpeta config y editaremos el archivo app.php para añadir la key, vermos que dicho código está comentado y además de descomentarlo, añadiremos la key (string) para que quede de esta forma:

`'key' => env('APP_KEY', base64_decode('aquí_la_key'))`
8. En las opciones del hosting iremos a “new database” y la crearemos con el nombre que queramos, el hosting nos dará un nombre de usuario y una clave codificados en base al nombre de la base de datos que hayamos proporcionado y nos lo dejará indicado junto a la base de datos creada.
9. Abriremos el gestor de bases de datos del hosting, en este caso phpmyadmin e importaremos la base de datos que exportamos de nuestro proyecto local.
10. Volveremos a abrir nuestro archivo .env y cumplimentaremos los datos de la parte de la base de datos con los que nos ha proporcionado el hosting, donde, DB_HOST será localhost según nos indica el hosting, DB_DATABASE el nombre que hayamos dado y, para DB_USERNAME y

DB_PASSWORD aquellos que nos generó el hosting. Modificaremos en este paso y en este archivo la constante APP_DEBUG y la pasaremos a false.

11. Verificar que todo es funcional y nuestro despliegue ya funciona en <https://gesticolmenar.000webhostapp.com/>

HERRAMIENTA DE APOYO GIT Y GITHUB

En el desarrollo de GESTICOLMENAR he hecho uso constante y exhaustivo de Git junto con GitHub para poder tener siempre hecho un seguimiento de los cambios realizados en el código del proyecto, así mismo, he hecho uso de la posibilidad de tener un repositorio remoto como GitHub para también mantener almacenado todo de forma segura, incluyendo el desarrollo de la memoria y demás elementos que han conformado esta aplicación web.

Dado que el uso ha sido únicamente por mi persona y no he visto en ningún momento necesario realizar el uso de funcionalidades como las ramas, se ha llevado a cabo todo el proyecto en la rama principal.

En resumen, el uso de Git junto con GitHub es una herramienta muy útil para cualquier proyecto de programación, pudiendo en caso necesario, realizar ramas o revertir cambios a versiones anteriores de tu código.

CONCLUSIONES

Conclusiones sobre el trabajo realizado

Las conclusiones finales con respecto al desarrollo serán las siguientes según me he ido encontrando con los distintos problemas.

- El pensar sobre una base de datos que será aplicada como uno de los pilares fundamentales de la aplicación requiere de un esfuerzo tremendo de imaginar posibles caminos, datos y relaciones a futuro, dando como resultado, un diseño de una base de datos bastante preciso al final, pero que, desde mi punto de vista, es inevitable ver carencias y deficiencias en el momento empiezas a avanzar en el desarrollo y, sin lugar a dudas, es una gran dificultad remodelar cualquier cosa en la base de datos por todos los “daños colaterales” que ello conlleva en el código, para mí, ha sido, uno de los puntos más complejos a llevar a cabo.
- Los mockups fueron realizados a modo de esbozo en formato papel, siendo otro gran punto de partida y a su vez otro punto con modificaciones con respecto a los modelos iniciales planteados, ya que, en el momento requieres de añadir cualquier otra funcionalidad o quitarla, muchas partes de los diseños han de ser revisados un por uno.
- El uso de Laravel ha facilitado, desde mi punto de vista, el montar todo el sistema de la App, ya que, de otro modo, hubiera sido tremadamente tedioso y estoy plenamente convencido llegado este punto en el que me encuentro redactando estas líneas y tras la desproporcionada cantidad de horas empleadas, es que, desde luego, no se hubiera podido llevar a cabo sin dicho framework.
- Laravel nos proporciona Eloquent, un ORM (Object-Relational Mapping) que nos permite interactuar con la base de datos de una manera más sencilla e intuitiva, es, a priori lo que pensé y, he de decir, que sí, es cierto, pero también es cierto que realizar consultas SQL que tengan cierta complejidad, me parecen un tanto menos intuitivas quizás por tratar de facilitar dicha sintaxis, acabas sin saber muy bien cómo formular dichas consultas, así mismo, la documentación no me ha parecido en este punto, nada bien estructurada ni sencilla de entender.

- A la hora de investigar un poco el mercado, las aplicaciones existentes y demás, no ha sido difícil darse cuenta de algo, que, inconscientemente sobreentendí y que, ha sido así, no existen casi aplicaciones en este campo y todas ellas manejan el mismo tipo de información que, es, en gran medida y por experiencia propia, la misma que manejan los apicultores en su día a día. Por ende, no ha sido un punto demasiado complejo de hacer, aunque, sí ha sido largo en lectura y búsqueda de posibles fuentes de financiación y poder realizar una análisis DAFO en condiciones y que se ajustara a la realidad.
- La parte del front, llevada a cabo con el motor de plantillas Blade de Laravel junto con el framework Bootstrap, dando pocos, pero algún pequeño matiz con CSS, ha sido una labor también muy larga y no falta de complicaciones debido a que, se ha hecho que fuera responsive gracias a Bootstrap, pero, a pesar de ello y contando con dicha herramienta, han existido situaciones difíciles de resolver para que quedara bien en cualquier tamaño de pantalla. Por otra parte, Blade facilita enormemente la faena de poder trasladar datos a las vistas.
- Con esta propia memoria he encontrado que ha sido necesario revisar bastantes elementos del curso pasado para poderla llevar a cabo, puesto que toda la parte de diseño de la base de datos, el paso a tablas, casos de uso... etc. han sido elementos que han requerido de un repaso previo para comenzar el proyecto algunos de ellos, y otros para poder realizar algunos pasos de esta memoria.

Conclusiones personales

Bueno, como conclusión personal, diré que el proyecto totalmente en solitario, ha sido un reto complejo y frustrante en muchas ocasiones y, en compensación y en partes más o menos igualadas, ha sido también satisfactorio y con partes más llevaderas que daban esos momentos de empuje necesarios.

El hacer un proyecto final, da una visión y perspectiva mucho más fidedigna de lo que sería un proyecto real, encontrando y viendo cada uno de los pasos a seguir, desde el papel en blanco y el IDE en blanco hasta empezar a darle forma a las ideas y empezar a “picar” las primeras líneas de código.

Como mencionaba, han existido momentos complejos ante errores difíciles, o bien de encontrar o bien de subsanar, dificultades a la hora de realizar ciertas funcionalidades, pero, lo que de verdad diría como conclusión final es que...

todos estos picos y valles han hecho una travesía con un final gratificante en lo personal, viendo, finalmente, cómo algo que sólo existía en mi mente, ha sido finalmente plasmado y visible para cualquier otro humano.

Posibles ampliaciones

He de decir que existen infinitos caminos y posibles ampliaciones, pero, si tuviera que decantarme por algunas que aportaran valor a GESTICOLMENAR, serían:

- Posibilidad de conexión de dispositivos GPS a la aplicación para la geolocalización constante de cada una de las colmenas, puesto que dichos dispositivos existen y son ampliamente comercializados en el sector por la gran cantidad de robos que se producen de colmenas.
- También me gustaría dejar que el usuario pudiera documentar visualmente lo que quisiera con, evidentemente, limitaciones de memoria, ya que, quizás quisiera guardar imágenes de posibles enfermedades que desconociera, plantas melíferas que no había visto, vídeos de ciertos elementos... etc.
- La posibilidad de generar informes PDF sería algo que, es posible fuera de bastante interés por parte de los apicultores, ya que, llegado final de año, quizás sería más conveniente tener diferentes reportes almacenados y descargables.
- El envío de emails con próximas visitas o recordatorios podría ser otra mejora interesante, sin abusar de dicho formato, dejando que fuera el usuario quien decidiera frecuencia y datos que quisiera recibir.
- Finalmente, otra propuesta podría ser la de añadir más campos a la base de datos, tales como el precio por kg según el tipo de miel y año, lo mismo para otros productos de la colmena, haciendo con ello una recolecta de datos interesantes.

BIBLIOGRAFÍA

Libros, artículos y apuntes

- Apuntes de Bases de Datos curso 2021-2022 de Sergio Badal, CEEDCV: he hecho uso de los apuntes del año pasado puesto que es una materia extensa y compleja y de la que no me acordaba de muchos puntos, por ende, he tenido que ir a releer y revisar muchos puntos del curso pasado para poder llevar a cabo el diseño de la base de datos.
- Apuntes de Entornos de Desarrollo 2021-2022 de Sergio Badal, CEEDCV: los apuntes para el desarrollo de los diseños de casos de uso han sido de vital importancia porque recordar todos los tipos de diseños es tarea imposible, y, por lo tanto, he ido a revisar dichos apuntes.
- Apuntes de Desarrollo Web Entorno Servidor 2022-2023 de Guillermo Garrido, CEEDCV: la aplicación se ha basado en Laravel y, los apuntes de este curso han sido completamente necesarios para la consulta de dudas con respecto al framework.

Direcciones web

- **getbootstrap.com** Dirección web del framework Bootstrap, se ha consultado en multitud de ocasiones durante todo el desarrollo de la aplicación. Además de para el uso de los CDN.
- **laravel.com** Dirección web del framework Laravel, se ha consultado en multitud de ocasiones durante todo el desarrollo de la aplicación.
- **<https://www.php.net/manual/es/index.php>** Manual del PHP empleado en alguna ocasión para buscar métodos o sintaxis que desconocía.
- **<https://blog.logrocket.com/exploring-best-laravel-chart-libraries/>** Blog donde se explican muchas de las librerías que existen para Laravel y generar gráficos, en ella se pudo ver cuál era la más optima para el desarrollo.
- **<https://aprendible.com/series/aprende-laravel-intermedio/lecciones/como-impedir-la-creacion-de-2-registros-identicos>** Web

con un curso de ciertos recurso específicos de Laravel que fue fundamental para poder lidiar con un problema que tuve, que, consistía en poder tener registros duplicados en la base de datos, pero que fueran únicos por usuario, ya que, si ponía como campo único en la base de datos, otro usuario no podría poner por ejemplo el mismo identificador a una colmena, algo que podría suceder, con esta explicación pude solventar el problema.

- **<https://www.laraveltip.com/como-crear-reglas-de-validacion-personalizadas/>** Explicación que me ayudó a solventar otro problema, y es que, Laravel cuenta con muchísimas validaciones ya generadas y de muy sencillo uso, pero, por ejemplo, no cuenta con validaciones para un DNI o un Código Postal, por ello, tuve que ver de qué manera se podía solucionar.
- **<https://coolors.co/image-picker>** Web usada para poder extraer la paleta de colores del logotipo empleado en la aplicación.
- **<https://publicdomainvectors.org/es/vectoriales-gratuitas/Abeja-de-dibujos-animados-con-miel/74973.html>** Logotipo de la aplicación de dominio público sin licencias y de libre uso.
- **<https://www.canva.com/>** Web empleada para poder confeccionar los mockups de la aplicación.
- **<https://fontawesome.com/>** De esta página se han usado todos los iconos empleados.
- **[Wikipedia.org](#)** Se ha consultado información sobre abejas, enfermedades, apicultura...
- **<https://www.ecocolmena.org/>** Me he documentado un poco también en esta web sobre apicultura.
- **<https://github.com/ndeet/unzipper/blob/master/unzipper.php>** Link a archivo php usado para el proceso de despliegue en 000webhost.