MIGRACIÓN DE WORDPRESS A SERVICIO CLOUD

INGENIERÍA CIVIL INFORMÁTICA

JUAN PABLO VERSALOVIC

ENERO DE 2023

1. Resumen ejecutivo

1.1. Español

El proyecto se llevó a cabo en la empresa "Yo Me Controlo", se decidió migrar la página a un servicio Cloud más seguro y privado. Los objetivos del proyecto incluyen garantizar un servicio seguro y privado cumpliendo con la ley HIPAA, evaluar el rendimiento de la página y desarrollar un backend estable.

Se exploraron diferentes opciones para el desarrollo web, incluyendo plataformas low-code, contratación externa y desarrollo propio. La opción elegida fue una combinación de JavaScript y Python, priorizando la seguridad y el conocimiento previo de los desarrolladores.

La metodología de desarrollo del proyecto se dividió en la creación de un MVP, la implementación de la orden médica, la búsqueda de laboratorio, y el método de pago y entrega de boletas. Se priorizó la seguridad mediante la encriptación de datos y el uso de servicios en la nube, específicamente Azure.

Se establecieron medidas de desempeño para evaluar el rendimiento y la seguridad, el rendimiento se evaluará mediante pruebas de esfuerzo, midiendo el tiempo de respuesta de la API. La seguridad se verificará mediante la encriptación de datos y el cumplimiento de la ley HIPAA.

El desarrollo del proyecto incluyó la creación de un MVP, implementación de la orden médica, método de pago, boletas, y un buscador de laboratorio. La conexión con servicios externos se gestionó mediante API y se implementaron medidas de seguridad como encriptación y almacenamiento seguro de claves. La página se encuentra en producción en Azure, aunque la implementación del firewall está pendiente por razones presupuestarias.

Las pruebas de esfuerzo indican que el sistema mantiene tiempos de respuesta aceptables, incluso con aumentos de usuarios. Se garantiza un rendimiento por debajo de los 6 segundos, que es el límite crítico para una API. Cuatro riesgos fueron identificados: pérdida de control de acceso, fallas criptográficas, componentes vulnerables y desactualizados, y atrasos en el desarrollo. Sus mitigaciones fueron desarrolladas. A pesar de desafíos en la comunicación con la empresa, la implementación mejoró la seguridad del servicio. Se sugiere aumentar el CPU para mejorar aún más el rendimiento, habiendo detectado un ligero impacto en la prueba de spike.

1.2. Inglés

The project was developed at "Yo Me Controlo". Given the operation with sensitive medical data, a decision was made to migrate the page to a more secure and private Cloud service.

Project objectives included ensuring a secure and private service in compliance with HIPAA regulations, evaluating page performance, and developing a stable backend. Various web development options were explored, with the chosen approach being a combination of JavaScript and Python, prioritizing security and developers' existing knowledge.

The project development methodology involved creating an MVP, implementing medical orders, laboratory search page, and establishing payment and invoicing methods. Security was prioritized through data encryption and the use of cloud services, specifically Azure.

Performance measures were established to evaluate performance and security. Performance would be assessed through stress tests, measuring API response times. Security would be verified through data encryption and compliance with HIPAA regulations.

Project development involved creating an MVP, implementing medical orders, payment methods, invoices, and a laboratory search feature. External service connections were managed through APIs, and security measures such as encryption and secure key storage were implemented. The page is in production on Azure, with firewall implementation pending due to budgetary reasons.

Stress tests indicate that the system maintains acceptable response times, even with increased users, guaranteeing performance below the critical 6-second threshold for an API.

Four risks were identified: Broken access control, cryptographic failures, vulnerable and outdated components, and development delays. Mitigations were developed for each.

Despite communication challenges with the company, the implementation enhanced service security. Increasing the CPU is suggested for further performance improvement, considering a slight impact detected in the spike test.

Contenido

2. Introducción

2.1. Contexto de la empresa

El proyecto a continuación será desarrollado en la empresa Yo Me Controlo, con orientación por parte de la Empresa Blueng Ingenieros SpA. Yo Me Controlo es una empresa de desarrollo de software que busca revolucionar la atención médica personal. Esto lo hacen con una plataforma de salud digital inteligente que indica los exámenes preventivos que se debería hacer el paciente según sus antecedentes personales y familiares y les genera una orden médica. Esto con el fin de eliminar el proceso de tener que ir a un médico para obtener dicho documento, y así ir directo al médico con los exámenes ya realizados y empezar con el tratamiento lo más antes posible si es que es necesario.

La empresa busca educar a las personas sobre la importancia de hacerse exámenes preventivos, estos últimos tienen como fin "prevenir algunas enfermedades a partir de la detección temprana de ciertos factores de riesgo para su desarrollo, y en caso de padecer alguna enfermedad, tratarla de manera oportuna y evitar posibles complicaciones" (Red salud 2022). Algunos ejemplos de enfermedades detectadas por medicina preventiva son: diabetes, hipertensión, cáncer, enfermedades cardiacas, entre otras.

La empresa fue parte del concurso Avante 03 de la armada de chile, en el cual obtuvieron primer lugar ganando el financiamiento para desarrollar la plataforma de Yo Me Controlo con la condición de proporcionar el mismo servicio a la armada, el enfoque del proyecto va dirigido a Yo Me Controlo no a la página de la armada, pero no hace falta decir que la pasantía se tuvo que desarrollar en ambas páginas, simultáneamente, lo que afectó el tiempo de desarrollo.

2.2. Problemática y proyecto

En la empresa, el enfoque del trabajo es en Backend , y como parte del trabajo solicitaron establecer un método de comunicación. El método que propuso la empresa era simplemente un correo de contacto, y una página de preguntas frecuentes. Esto haría que no se logre responder a preguntas específicas que el usuario tenga de algún antecedente o dato médico personal, sin tener que enviar un correo con información sensible.

Sobre esto, un consejo entregado por Ubisend para las empresas es: "Businesses need to expand their concept of reachable. Being available outside of traditional email and phone now has a financial upside"(Ubisend 2021 p. 13). Lo que dice este consejo es que empresas deberían expandir su concepto de

conectividad, siendo capaces de diferenciarse del típico correo y teléfono, además de obtener ventajas financieras.

Por lo que se propuso si fuese mejor algún sistema más autónomo, a partir de eso se planteó hacer un chat-bot con IA (Inteligencia Artificial) que proveería una estable y segura comunicación con los clientes en la página de Yo Me Controlo.

Se investigo el uso de chat-bot con IA para servicio al cliente y se encontró que la empresa Ubisend generó un documento con las estadísticas sobre el uso estos. La empresa muestra que un 55% de los entrevistados son más propensos a hacer negocios con compañías que puedan escribirles y que el 98% de clientes piensan que los chat-bot entregan un beneficio al servicio de cliente.

Al momento de exponer la idea de la creación de dicho chat-bot para la página la empresa no lo encontró prioridad, en cambio propusieron resolver un problema enfocado en la página web, pues esta tiene mayor relevancia. Como Yo Me Controlo tenía una página en WordPress, que se encontraba temporalmente fuera de servicio, se presentó una reestructuración de la página.

WordPress es un content managment service (CMS), software usado para crear, manejar, y modificar páginas web sin la necesidad de saber programar. Este es el más usado en el mundo, tomando un 63% del mercado, que equivale a un 43% de las páginas web del mundo (W3Techs 2023). Pero la comodidad de usar WordPress es pagada por su vulnerabilidad, ya que se basa en el uso exclusivo de plugins, componentes que generan una rápida integración sin la necesidad de programarlos. Dado que estos plugins son reutilizados por todos los usuarios de WordPress, cualquier vulnerabilidad encontrada en un determinado plugin puede ser explotada para atacar cualquier página que utilice esa versión.

La empresa de ciberseguridad Sucuri, dedicada a la detección de vulnerabilidades en páginas Web, en su último reporte muestra la distribución de páginas infectadas. En este encontraron que WordPress abarcaba el 96.2% de todas las páginas, ver gráfico 1, con esta información se confirma que, al ser uno de los CMS más populares, este se ve afectado más a menudo.

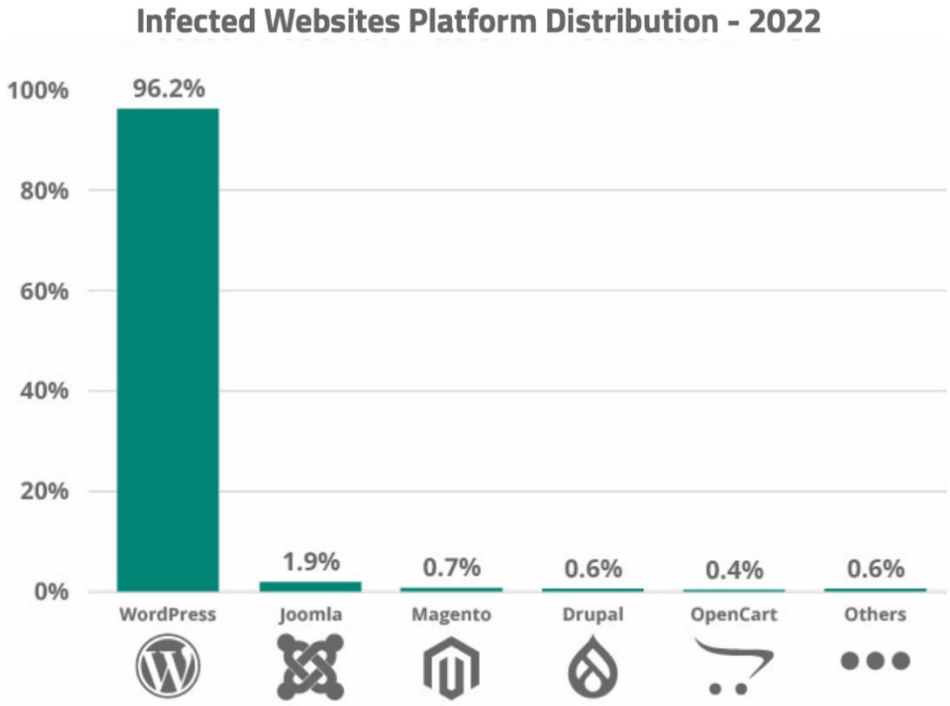

Gráfico 1 Distribución de plataformas de sitios web infectados.

La operación con datos personales y médicos sensibles del usuario puede dar lugar a problemas de seguridad. Algunos ejemplos de ataques maliciosos incluyen la obtención de datos bancarios durante el proceso de pago, el robo de identidad, entre otros.

Además, Wordfence, empresa de ciberseguridad dedicada en la detección de ataques en WordPress, indica en último reporte de amenazas 2020 que hubieron más de 90 mil millones de ataques maliciosos. Se registraron más de 57 millones de direcciones IP[1] distintas, lo que equivale a 2,800 ataques por segundos (Gall 2021). Dado que WordPress presenta vulnerabilidades de seguridad, se tomó la decisión de llevar a cabo el Proyecto de migración de la página web desde WordPress hacia un servicio Cloud[2] más seguro y privado.

[1] Estas son direcciones únicas que se utilizan para identificar a un dispositivo.

[2] Es un servicio que permite el almacenamiento y acceso a servicios informáticos atreves de internet sin la necesidad de un lugar físico.

3. Objetivos

El primer objetivo de este proyecto es proporcionar un servicio seguro y privado a los clientes, esto por la ausencia de seguridad que tenía la página, WordPress no tomaba ninguna medida de encriptación de los datos médicos. Para saber cómo se tiene que tratar los datos médicos se buscó una ley para regirse en cómo hacer un buen trato de los datos. Cómo no se encontró una chilena se optó la ley HIPAA, esta es "la ley de transferencia y responsabilidad de los seguros médicos de EE. UU. o HIPAA *(Health Insurance Portability and Accountability Act*), de 1996, es una ley federal que establece requisitos de privacidad y seguridad de los datos para las organizaciones que deben salvaguardar la información médica protegida de otras personas" (Google Cloud, s.f.).

Para cumplir con la ley se tiene que encriptar los datos del usuario y usar un servidor que sea HIPAA conforme. La empresa ya tiene un convenio con Azure[3] que califica en los estándares HIPAA. Para medir si cumplen el objetivo se revisarán los estándares de la ley HIPAA. Si se cumplen el 100% de ellos, los datos se categorizarán como seguros y privados.

El segundo objetivo es respecto al esfuerzo de la página, esto se refiere a cuántos usuarios podrá aguantar la página y cuál es su rendimiento en distintos escenarios. Esto importante para saber cuándo es necesario aumentar la potencia del servidor y que así aguante los nuevos usuarios en momentos que se sabe que va a haber un aumento de flujo, esto normalmente ocurre cuando se implementa una oferta como el *Cyber Monday* o el *Black Friday* que son eventos de ofertas masivas que atraen un alto tráfico a sitios web.

Por otro lado, para medir si el objetivo de esfuerzo es cumplido se usará la prueba de esfuerzo. La página Loadview dice que una prueba de esfuerzo se utiliza para "determinar la confiabilidad y estabilidad de todos sus recursos web, como sitios web, aplicaciones y API. Las pruebas de estrés tienen como objetivo encontrar el punto de ruptura de un sitio web / aplicación bajo una carga extremadamente alta durante un período de tiempo" (LoadView, 2020) Para esto se usan tres tipo de pruebas de esfuerzo, primero se tiene que hacer la prueba de esfuerzo promedio para saber el rendimiento de la página con una cantidad normal de usuarios y así tener una base para comparar, después se puede hacer la de estrés que mide el rendimiento cuando se aumenta por encima de la cantidad normal usuarios, y por último la prueba de *spike* que mide la estabilidad de la página cuando haya un gran flujo de usuarios en un tiempo muy corto.

[3] Es un servicio de nube creado por Microsoft que te deja compilar, implementar y administrar aplicaciones.

Por último, el objetivo *Smart* es desarrollar, antes de que termine el año, un backend para la página de yo me controlo para lograr un funcionamiento estable y una conexión segura con los clientes. Este será medido según el esfuerzo y qué tan seguro y privado según la ley HIPAA.

4. Estado del arte

Para el desarrollo web, en términos generales, existen tres grandes formas de hacerlo: hacer uso de una plataforma low-code, que no requiere o necesita muy poca experiencia de programación; externalizar el proceso, es decir, contratar una empresa externa para el desarrollo de la página web; o bien, programarla desde cero.

En la primera opción son plataformas que se dedican simplificar el proceso de desarrollo web, esto se logra estandarizando los procesos rutinarios para la creación de una página. Un ejemplo son los plugins de WordPress. Otra practica utilizada es proveer al cliente con temas de diseño predefinidos a la categoría de página que se trata de crear, por ejemplo, si se requiere de una página para vender productos el diseño ofrecido contendrá un sistema de carrito de compra, pero si uno desea crear un blog el diseño ofrecerá cuadros de texto y no un carrito.

Estas plataformas entregan un sistema de seguridad según el nivel de suscripción, al aumentar el precio entregan mayor seguridad. En WordPress, de los cincos niveles de suscripción, solo los dos últimos entregan una web app firewall, que se usa para bloquear los ataques maliciosos como *SQL injection* o *Cross-site scripting*. Otro servicio de seguridad entregado solo en estos dos niveles es el aislamiento de la infraestructura, es decir, se alojará[4] en un servidor privado. Esto es un gran beneficio, ya que tener un servidor compartido afecta el rendimiento y limita los recursos entregados.

La segunda opción es contratar una empresa para la creación de la página web, esta se encargaría de la seguridad, privacidad, funcionamiento, y diseño. Al momento de tomar una decisión se debe tener en consideración una empresa que haya trabajado con HIPAA o que tenga desarrolladores entrenados en seguridad.

La tercera opción es la creación propia de la página desde cero usando distintos lenguajes de programación y alojándola con una empresa para poder levantar la página al público. Una ventaja es poder manejar la seguridad de la página por medio de utilizar solo componentes seguros, probar la seguridad de la página, y arreglarla con facilidad.

[4] Es entregar un servicio de almacenamiento de datos que se puedan visualizar en internet

5. Propuestas de solución

Las propuestas de solución son: Primero, continuar con la página de WordPress, pero alojarla en Azure, y desarrollar la seguridad necesaria para que sea conforme HIPAA. Segundo, usar JavaScript con Reactjs para la interfaz del usuario y NodeJs para el servidor, y se alojará en Azure. Cuarto, con Python se usará Django para la interfaz del usuario y Flask-SQLAlchemy para desarrollar el backend, también se alojará en Azure. Y, por último, está la opción de externalizar el proceso, es decir, buscar un servicio que les desarrolle una web App desde cero, y que cumpla con la ley HIPAA.

5.1. WordPress enfocado en seguridad

Hipaajournal es una empresa dedicada a listar los requerimientos para que una página cumpla con los estándares HIPAA, los que solo afectan a WordPress son: la necesidad de tener solo plugin que sean confiable y que además sean compatibles con la última versión de WordPress. Otra forma de seguridad que se debe tomar es que el anfitrión de la página sea un servidor HIPAA conforme y, además, si se van a guardar datos de salud deben ser fuera de WordPress.

5.2. JavaScript

Lenguaje de programación orientado a objetos. La página de Amazon dice: “En la actualidad, puede utilizar JavaScript para el desarrollo tanto del lado del cliente como del lado del servidor” por eso, en esta propuesta, se tiene planeado usar la biblioteca[5] Reactjs para generar la interfaz de usuario de la página web y, por el lado del servidor, se usará NodeJs y Express para crear una API[6] que se conectará a la base de datos en Azure.

5.3. Python

Lenguaje de programación multiparadigma que es más usado para el backend al momento de desarrollo web, la página de Amazon dice que: “Python es útil para escribir código del lado del servidor debido a que ofrece muchas bibliotecas que constan de código pre escrito para crear funciones de backend complejas” (Amazon). Por lo que, en esta propuesta, se usaría la biblioteca de Django para la creación de la interfaz del usuario y la biblioteca Flask para la conexión a la Base De Datos en Azure.

[5] Conjunto de código utilizado para el desarrollo.

[6] Software que permite la comunicación entre dos aplicaciones.

5.4. Externalizar el proceso

Se tendrá que buscar una compañía dedicada a la creación de páginas web con experiencia en seguridad para lograr cumplir los estándares HIPAA, tiene que ser menor al costo de los desarrolladores que se están ya pagando y poder cumplir con los diseños pedidos por la empresa y diseñador.

6. Solución escogida

Al momento de escoger la solución se tomaron en consideración la seguridad de la página y el conocimiento de los desarrolladores trabajando en esta. No se escogió WordPress, ya que para volverlo HIPAA conforme se requiere de una buena mantención y seguridad lo que requiere de especialistas en seguridad en WordPress y los desarrolladores contratados no se especializan en ese tema. Sobre esto, la empresa Hipaajournal dice:

> *"Our recommendation is to develop your own website from scratch that is easier to secure and maintain, host the site with a HIPAA compliant hosing company, and if you do not have employees with the correct skill sets, use a vendor that specializes in developing HIPAA compliant websites and patient portals." (Alder, 2022).*

Es decir, recomienda que es mucho mejor crear una página de cero, y si se quiere usar WordPress, es posible hacerlo, pero se necesita a trabajadores expertos en seguridad para lograrlo.

Tampoco se escogió la solución de externalizar la página, ya que estas suelen ser estándares con poco rango de customización, por lo que no se lograría ajustar a los estándares pedidos por la empresa y el diseñador. Otro problema que se vio fue la seguridad que ofrecían, era demasiado básica y no cumplía con los estándares HIPAA de seguridad, por lo que se descartó.

Se escogió desarrollar la página con Python y JavaScript pues los desarrolladores ya se manejaban con estos lenguajes. El problema fue que, al momento de implementar Python en el backend, ocurrieron complicaciones, por lo que se cambió por JavaScript. De igual modo, el equipo a cargo solicitó no descartar Python del todo, porque se podía usar para la parte de Machine Learning (ML).

7. Evaluación económica.

Para este proyecto, se hizo una evaluación económica[7] con proyección a 5 años, en esta se tomaron como supuesto que todos los usuarios al menos compraron un examen. En el escenario pesimista se asume que cada año solo se aumenta en 24 usuarios cada mes. Y en el escenario optimista que cada año se duplica la cantidad de usuarios que hubo en el año anterior.

Con estos supuestos, se obtienen los siguientes resultados: en el escenario pesimista, el VAN es de 189 mil 900 pesos con una TIR del 6 por ciento, mientras que, en el escenario optimista, el VAN es de 150 millones, 179 mil 548 pesos con una TIR del 123 por ciento.

En conclusión, en ambos escenarios, la VAN fue positiva, independiente de que el aumento de flujo de personas aumente los costos de mantener la página; pero, si el flujo es menor que 24 usuarios mensuales al año, la VAN es negativa, por lo que no sería rentable el proyecto. Por lo tanto, se recomienda buscar una empresa para generar un convenio y poder atraer más clientes. Esto se vio en el tiempo de la beta, donde se realizó un convenio con la empresa Farmacia Fracción y se observó un aumento de usuarios. Todos los datos para el flujo de caja encuentran en el anexo 13.2.

[7] Tablas de flujo de caja: https://docs.google.com/spreadsheets/d/1LaD-0Zhd5126a45xXgZgxhsZbeujbs_-/edit?usp=sharing&ouid=110805920538660969589&rtpof=true&sd=true

8. Metodología

Para lograr la migración de WordPress a un servicio cloud desarrollado en JavaScript se deberá contar con las mismas funcionalidades que tenían en la página web anterior. Estas son: un inicio de sesión, un registro de usuario, preguntas para el usuario, método de pago, entrega de boletas por correo electrónico, creación de la orden médica en formato PDF, y un buscador de laboratorio.

El equipo de Front-end, a partir del trabajo del diseñador, usará Reactjs para crear los diseños necesarios en el front-end, como parte del proceso de migración de WordPress a un servicio en la nube basado en JavaScript.

En el backend se usará React-PDF para la creación de las órdenes médicas, para el sistema de inicio de sesión y registro se usará una API en NodeJs que va a estar conectado a la base de datos en Azure, y para el método de pago y boletas se contrató un servicio que tenga conexión por API.

Por último, el buscador de laboratorio se consiguió trabajando con la API de GoogleMaps y su plataforma *QuickBuilder*.

Para que sea seguro es necesario encriptar la información que entrega el usuario, guardar de forma segura las llaves de acceso a las API utilizadas, implementar un firewall, y tener todo levantado en Azure.

Se encriptó la información del usuario por medio de hashing[8], se guardaron las llaves de acceso a las API usando key-vault[9] de Azure, se usó Azure para implementar el firewall y por último se levantó todo en Azure usando la plataforma de GitHub que tiene una integración automática.

[8] Método criptográfico que transforma datos de cualquier tamaño en un valor hash.
[9] Es un servicio en la nube para almacenar secretos y poder accederlos de forma segura.

9. Medidas de desempeño

Para medir el objetivo de esfuerzo, específicamente el rendimiento, se usará el tiempo de respuesta de la API, es decir, cuánto se demora en recibir el pago. Se tomará como tiempo de respuesta exitoso si se logra tener una respuesta menor a 6 segundos en el percentil 95 en las tres pruebas de esfuerzo. La prueba de esfuerzo promedio se realizará con el promedio diario de usuarios de la beta, que equivale a 6 usuarios. Para la prueba de estrés, se aumentará a 18 usuarios, y en la prueba de spike se utilizarán 60 usuarios, que representa 10 veces más de lo esperado normalmente. Se aplicará un efecto aleatorio para cada iteración para simular el comportamiento del usuario en la página. La estabilidad de la página se evaluará asegurándose de que todas las pruebas de esfuerzo entreguen el código 200 que significa ok.

Para medir el objetivo de brindar un servicio seguro y privado se seguirá la ley HIPAA, para esto se necesitará encriptar los datos del usuario, alojar la página en un servidor que sea reconocido por HIPAA. Si se logra implementar, los datos se considerarán seguros, d.do que el resto del cumplimiento HIPAA se realizará por el equipo de frontend y el equipo de la base de datos.

10. Desarrollo del proyecto

Primero, la empresa necesitaba un MVP[10] así que se empezó con la creación de la orden médica. Para esto se usó la biblioteca de React-PDF y se usó una encuesta diseñada por el equipo de front-end para saber qué exámenes se tiene que hacer cada paciente. Se conectaron las variables al local storage para entregarlas al PDF sin necesitar de la base de datos. Esto se hizo con las funciones useContext y Context Provider[11] de React. Para entregar el MVP se usó un ambiente de prueba en Azure llamado Qa-env que consiste en una página web estática[12] donde se subió el repositorio[13] de GitHub.

Como no estaba completada la base de datos, se trabajó en el método de pago. Para esto se tomó en consideración usar una pasarela de pago, ya que esto agilizaría la programación de la página. Al momento de decidir cual implementar se revisó las distintas formas de pago que poseen, la documentación, y si contaban con ambiente de prueba. Al final se eligió Flow, ya que tenía una extensa documentación, contaba con ambiente de prueba, y tenía convenio con distintos medios de pago sin la necesidad de tener un acuerdo directo con cada uno de ellos.

Se utilizo una API rest[14] creada en NodeJs para conectar el medio de pago con la página, se usó como base la clase Flow creada por Esteban Fuentealba (2023) esta se dedica a ordenar los datos para enviar y entregar la encriptación que requiere Flow, ya que este pide encriptar los valores con HMAC que es una técnica de autentificación criptográfica que usa una función hash[15] en este caso SHA256 y una llave secreta esta la entregado Flow.

[10] Es un producto que contine los mínimos requerimiento y funcionalidades para que un software salga al mercado.
[11] Archivos con funcionalidades que generan el traspaso de información a través del almacenamiento local.
[12] Son páginas que entregan al usuario exactamente como se almacenaron, esto significa que se muestra la misma información a todos los usuarios.
[13] Es donde se almacena digitalmente el código y donde los desolladores pueden realizar cambios y administrar el código de una aplicación.
[14] Son un tipo de API que utilizan las funciones POST, GET, PUT, etc. Para generar la comunicación entre la aplicación y la base de datos.
[15] Es un algoritmo matemático que transforma cualquier bloque de datos en una serie de caracteres de un largo fijo.

```
65    sign(params) {
66      const keys = Object.keys(params)
67        .map(key => key)
68        .sort((a, b) => {
69          if (a > b) return 1;
70          else if (a < b) return -1;
71          return 0;
72        });
73      let toSign = [];
74      keys.map(key => {
75        toSign.push(key + "=" + params[key]);
76      });
77      toSign = toSign.join("&");
78
79      return CryptoJS.HmacSHA256(toSign, this.secretKey);
80    }
```

Código 1 obtenido de Esteban Fuentealba, genera la encriptación de los datos de la manera que Flow los pide. Elaboración propia.

Para enviar a la página se utiliza la función post[16] para enviar el URL que entrega Flow y poder redirigir al usuario al método de pago.

```
app.post("/apiFlow/create_order", async (req, res) => {
    try {
      // Prepara el arreglo de datos
      const serviceName = "payment/create";
      // Instancia la clase FlowApi
      const flowApi = new FlowApi(config);
      // Ejecuta el servicio

      let response = await flowApi.send(serviceName,req.body, "POST");

      //Prepara url para redireccionar el browser del pagador
      redirect = response.url + "?token=" + response.token;
      res.json({
        redirect
      });
    } catch (error) {
      res.json({ error: error.message });
    }
  });
```

Código 2 Envía los datos encriptados con el código1 a la API de Flow y espera la respuesta y la envía a la página de yo me controlo para poder ser redirigido a la pasarela de pago.

[16] Es la función que se utiliza para crear nuevos datos en la API.

Para las boletas se utilizó el sistema que la empresa ya tenía contratado, llamado Lioren. Para poder enviar las boletas se utilizó la librería Nodemailer a través de la API, esto ocurre justo después de recibir la confirmación de pago en Flow, revisar el anexo 15.4 para más profundidad.

Cuando estuvo lista la base de datos, se conectó al frontend mediante la API rest. Después se creó el inicio de sesión y registro de usuario, con enfoque en guardar los datos del usuario de forma segura. Para lograrlo se encriptaron las contraseñas con la técnica HMAC, que usa la función hash SHA512 y una llave secreta guardada en Azure key-vault.

Para el buscador de laboratorio se usó la API de GoogleMaps y la plataforma de QuickBuilder de Google, esta entrega un buscador de laboratorio que se puede incorporar por medio de un Iframe[17] directo a la página. Lo único que hay que hacer para que se vean los laboratorios es ingresarlos manualmente para generar los marcadores, por esto se colocaron solo los que se tenían convenio, que son los de IntegraMedica y RedSalud que se encuentran en Santiago.

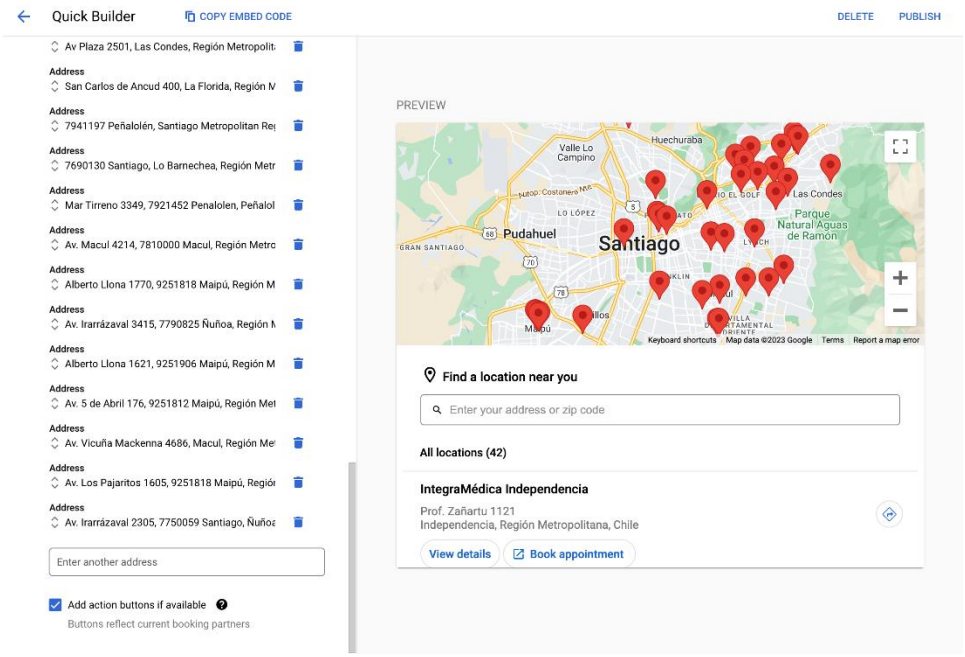

Imagen 1 plataforma de Google Quick Builder que se utilizó para la creación del buscador de laboratorio.

Por último, se subió a producción la API usando una web app, y se levantó la página usando una Static Web app. Ambos se levantaron en Azure. Los códigos fueron subidos a GitHub, cada uno en su repositorio correspondiente.

[17] Elemento HTML que permite insertar videos, documentos y código directo a la pagina.

En Azure se encuentra la API, la base de dato SQL, el Key-vault, el ambiente de pruebas dicho anteriormente, y la pagina en producción, ver imagen 2 de la arquitectura cloud.

No se implementó el firewall por falta de presupuesto, como tiene un costo de 285 dólares por mes se está esperando en el apruebo del presupuesto para su implementación más adelante.

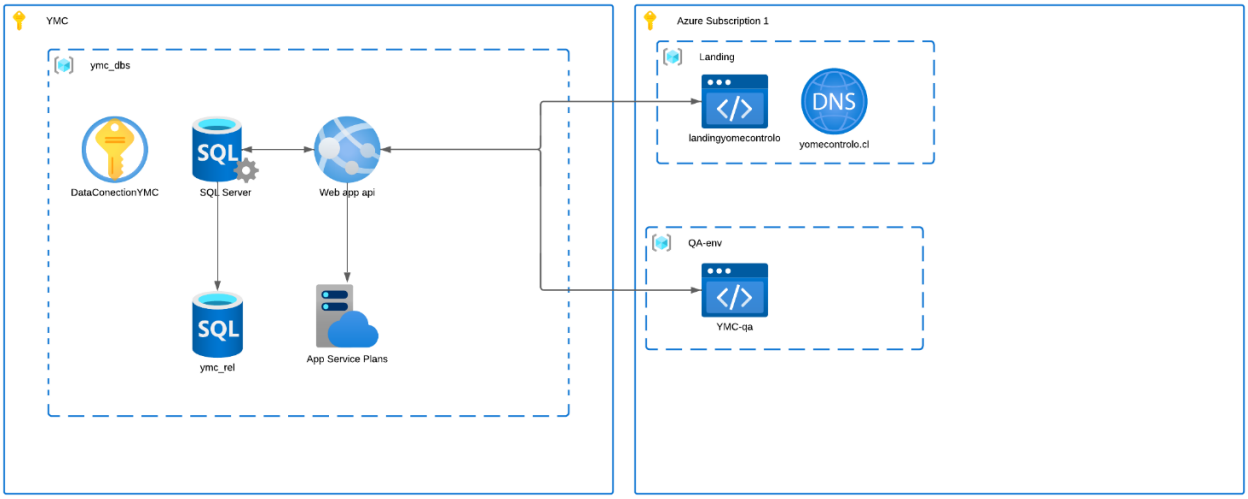

Imagen 2 la arquitectura cloud de Azure se puede ver como la api se encuentra en el medio de la base de datos y las páginas estáticas entregando y recibiendo información a la página de prueba YMC-qa y la de producción landingyomecontrolo. Elaboración propia.

11. Resultados

11.1. Prueba de esfuerzo promedio

En primer lugar, el percentil 95 de la prueba de esfuerzo promedio fue un tiempo de respuesta de dos segundos. Esto significa que 95% de los usuarios un tiempo de respuesta de dos segundos o menor, estos usuarios generaron un total de 618 preguntas al API con un promedio de 1687 milisegundos en el tiempo de respuesta. Todas las iteraciones respondieron con el código 200 calificando la prueba como aprobada.

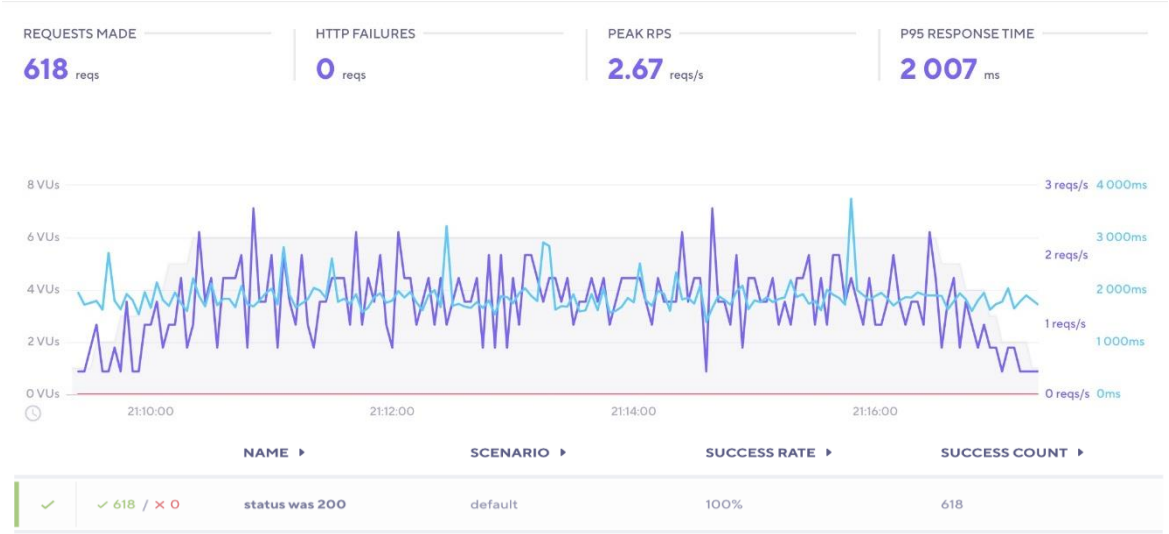

Imagen 3 Prueba de esfuerzo promedio se puede ver el resultado del percentil 95 fue de 2 segundos y que todas las iteraciones lograron entregar el código 200. Elaboración propia.

11.2. Prueba de estrés

En la prueba de estrés se puede ver que al aumentar los usuarios a 18 esto genera un aumento en el tiempo de respuesta de 152 milisegundos en el percentil 95, lo que significa que el 95% de los usuarios obtuvieron un tiempo de respuesta de 2.1 segundos o menor, se generaron 1803 preguntas a la API con un promedio de 1732 milisegundos en el tiempo de respuesta. Todas las iteraciones respondieron con el código 200 calificando la prueba como aprobada.

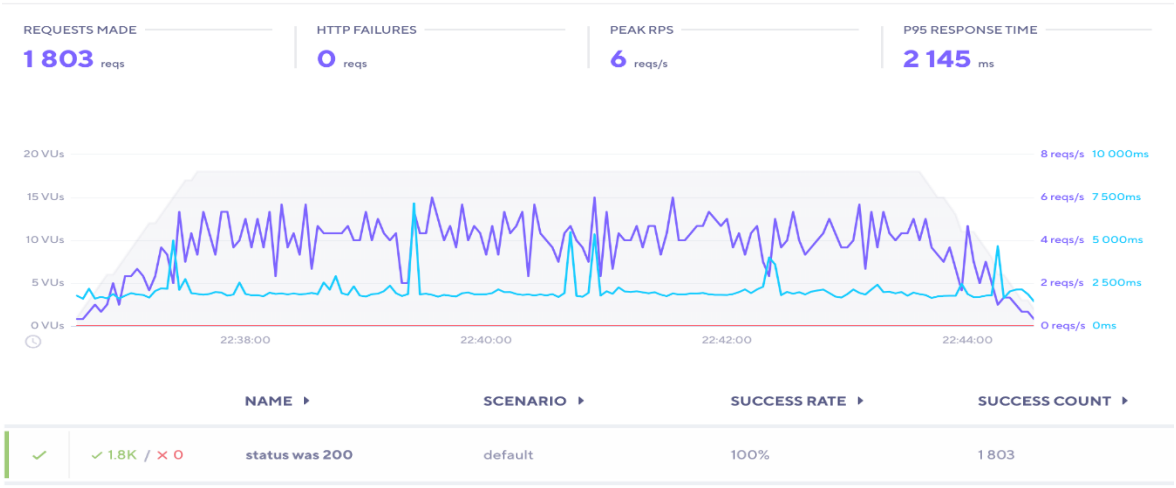

Imagen 4 prueba de estrés el tiempo de respuesta fue de 2.1 segundos en el percentil 95 y todas las iteraciones entregaron el código 200. Elaboración propia.

11.3. Prueba de spike

Por último, la prueba de spike se aumentaron a 60 usuarios, en el percentil 95 se obtuvo un tiempo de respuesta de 4443 milisegundos, significa que el 95% de los usuarios obtuvieron un tiempo de carga de 4.4 segundos o menor, se generaron 1215 preguntas a la API con un promedio de 2029 milisegundos de tiempo de respuesta. Todas las iteraciones devolvieron con el código 200 calificando la prueba como aprobado.

Imagen 5 prueba de spike el tiempo de respuesta fue de 4.4 segundos en el percentil 95 y todas las iteraciones entregaron el código 200. Elaboración propia.

11.4. Análisis de los resultados

Se puede ver que en todas las pruebas[18] entrego el código 200 y que independientemente si se aumenta los usuarios el rendimiento máximo que se puede observar es menor a 6 segundos que es el tiempo máximo que se puede demorar una API (Hamilton 2023), se tomó la decisión de que estaba bien los tiempos de respuestas, ya que no se espera más de 60 usuarios simultáneamente usando la página. Por si algún motivo el tiempo de respuesta se demoraría más de lo revisado se implementó que al momento de presionar el botón de pasar a pagar este se cambia a redirigiendo para que el usuario que debe esperar para ser redirigido a la página de pago de Flow se puede ver el botón en el anexo 13.5.

[18] Para ver las pruebas de esfuerzo en profundidad ver https://drive.google.com/drive/folders/1MnUfu5KTRzQkK7vUPcWvocX9M6XP8LDJ?usp=drive_link

12. Matriz de riesgos

Una vez el proyecto esté implementado puede ocurrir brechas de seguridad, así que para mitigar se hizo esta matriz de riesgo.

Se encontraron principalmente cuatro riesgos: Pérdida de control de acceso, fallas criptográficas, componentes vulnerables y desactualizados, y retraso del desarrollo. Para establecer su tasa de incidencia e impacto se usaron los factores provistos por la lista de top 10 vulnerabilidades 2021 de la fundación OWASP, Open-Source Foundation for Application Security. Esto se usó para generar la matriz (OWASP 2023).

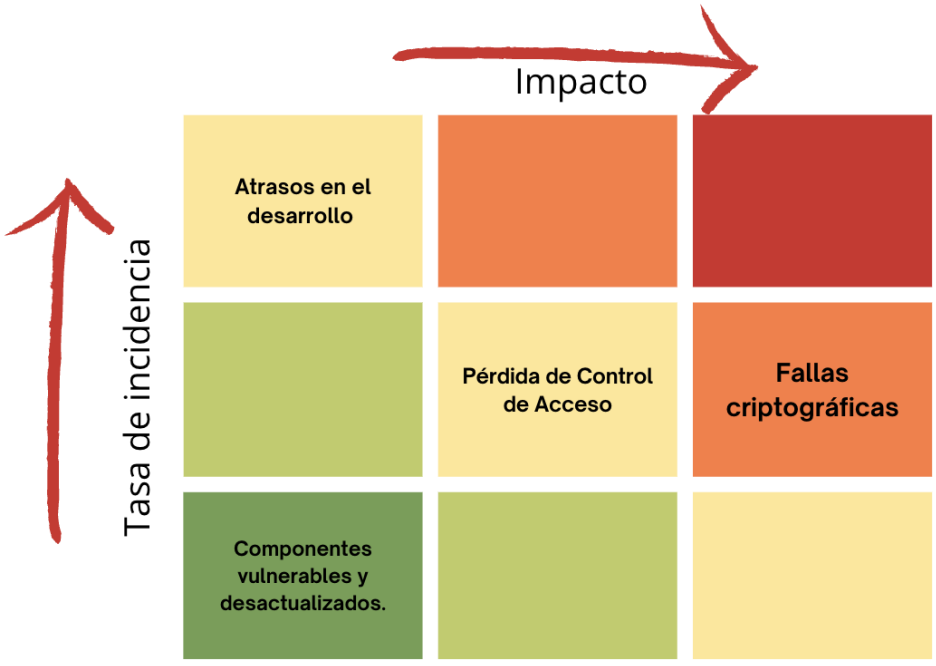

Imagen 6 matriz de riesgo donde se establecieron los cuatro posibles riesgos que son: pérdida de control de acceso, Fallas criptográficas, componentes vulnerables y desactualizado y atrasos en el desarrollo. Elaboración propia.

12.1. Pérdida de control de acceso:

Según los factores en la lista mencionada, el riesgo de pérdida de control de acceso tuvo una tasa de incidencia máxima de 55.97%, dejándolo en Incidencia media. El impacto calculado fue de 5.93, esto se categorizó como Impacto medio.

Respecto a este riesgo, la OWASP dice “El 94% de las aplicaciones fueron probadas para detectar algún tipo de pérdida de control de acceso con una tasa de incidencia promedio del 3,81%. Este problema tuvo la mayor cantidad de ocurrencias en el conjunto de datos analizado con más de 318.000” (OWASP 2023).

12.2. Fallas criptográficas:

Del mismo modo, el riesgo de fallas criptográficas tuvo una tasa de incidencia máxima de 46.44%, dejándolo en Incidencia media. El impacto calculado fue de 6.81, esto se categorizó como Impacto alto.

12.3. Componentes vulnerables y desactualizados:

Finalmente, el riesgo componentes vulnerables y desactualizados tuvo una tasa de incidencia máxima de 27.96%, dejándolo en Incidencia baja. El impacto calculado fue de 5.00, esto se categorizó como Impacto medio.

Sobre esto, OWASP comenta “Los componentes normalmente se ejecutan con los mismos privilegios que la propia aplicación, por lo que las fallas en cualquier componente pueden tener un impacto grave. Tales fallas pueden ser accidentales (por ejemplo, error de codificación) o intencionales (por ejemplo, una puerta trasera en un componente)” (OWASP 2023).

12.4. Atrasos en el desarrollo

El atraso en el desarrollo tiene una probabilidad alta, ya que es muy común que queden tareas sin realizar, si esto ocurre recurrentemente genera un problema, dado que se quedan tareas sin resolver o peor aún no se puede hacer las nuevas tareas por que dependen de las no realizadas, por eso el impacto es medio.

12.5. Mitigaciones

Para mitigar el primero se usó una API que maneja los accesos del usuario como el registrar e iniciar sesión o el redireccionamiento al método de pago, solo se pude acceder con peticiones especificas a la API y estas requieren de llaves secretas que se encuentra guardadas en la nube.

Para mitigar el segundo se usó un método de pago externo, de esta manera nunca se tiene acceso a sus credenciales, quitando así el riesgo de que alguien las robe.

Para mitigar la tercera se eliminó cualquier componente que no fuera imprescindible, estos se desarrollaron desde cero, y los componentes necesarios se mantienen siempre actualizados y en su última versión.

Para mitigar la última se utilizó la metodología ágil scrum se realizaron *sprint* de 2 semanas con reuniones entremedio de cada uno para restructurar el *sprint* y ver que tiene más o menor prioridad.

13. Conclusión

La mayor dificultad que se encontró fue la mala comunicación que se tenía con la empresa, ya que estos no sabían lo que realmente deseaban o necesitaban. Esto ocasionó situaciones en las que se comenzaba el *sprint* con tareas para la página de Yo Me Controlo porque desde un inicio se había establecido como prioridad número uno. Sin embargo, a mitad del *sprint*, el jefe de proyecto llegaba con noticias de la empresa indicando cambios de prioridad y la necesidad de avances urgentes en el proyecto de la armada. Lo bueno es que la mayoría de las veces se podía reutilizar el código que se estaba trabajando en Yo Me Controlo en la página de la armada. Esto afectó bastante el desarrollo de la solución, ya que obligaban a cambiar constantemente de repositorio. Esto provocó que el proyecto fuera bajado en la lista de prioridad, afectando el plan de implementación de la solución y dejando de lado la funcionalidad de iniciar y registrar usuarios. Esta funcionalidad fue desarrollada e implementada en el ambiente de prueba, pero fue bajada de prioridad por el mismo CEO, ya que se necesitaba apresurar el lanzamiento a producción para ir a trabajar en el proyecto de la armada.

Con la propuesta de solución se logró brindar un servicio más seguro del que tenían anteriormente, dado que se eliminó la presencia de plugin vulnerables, se alojó en un servidor cloud seguro “Azure” y se encripto los datos del usuario por medio de hash, lo único que falto fue la implementación del firewall que quedo en manos de la empresa tomar la decisión de implementar.

Por parte del rendimiento se observó un aumento en la estabilidad de la página, ya que en las pruebas realizadas todas devolvieron el código 200 y que el tiempo de respuesta promedio de todas las pruebas se encuentra alrededor de los 1.5 segundos hasta 2 en el peor escenario que eso todavía califica por debajo del máximo de 6 segundos que es el tiempo máximo de demora en una API. Para lograr mejores tiempos de respuesta se recomienda aumentar el CPU, dado que al momento de realizar las pruebas se detectó que el CPU llego a 70% en la prueba de spike por lo que si vio afectado el rendimiento.

14. Bibliografía

Alder, S. (2022, 3 de noviembre). Can You Make WordPress HIPAA Compliant? HIPAA Journal. Recuperado de https://www.hipaajournal.com/wordpress-hipaa-compliant/

Amazon Web Services. (2023). What is Python? Recuperado de https://aws.amazon.com/es/what-is/python/

Amazon Web Services. (2023). What is JavaScript? Recuperado de https://aws.amazon.com/es/what-is/javascript/

Autoadministrables. (2023). Diseño de páginas web en Chile. Accedido 24 noviembre de 2023 Recuperado de https://www.autoadministrables.cl

Clarity Ventures. (2023, 20 de octubre). WooCommerce HIPAA Integration: Is WordPress HIPAA Compliant? How to Keep WordPress & WooCommerce Secure. Recuperado de https://www.clarity-ventures.com/woocommerce-hipaa-ecommerce-integration/hipaa-compliance-for-wordpress-woocommerce

Fuentealba, E. (2023, abril 29). flowcl-node-api-client. GitHub. Recuperado 6 de diciembre 2023 de https://github.com/EstebanFuentealba/flowcl-node-api-client

Gall, R. (2021, 27 de enero). The Wordfence 2020 WordPress Threat Report. Wordfence. Recuperado 3 diciembre 2023 de https://www.wordfence.com/blog/2021/01/the-wordfence-2020-wordpress-threat-report/

Google Cloud. (s.f.). HIPAA Compliance. Recuperado el 2 de diciembre de 2023, de https://cloud.google.com/security/compliance/hipaa-compliance?hl=es

Hamilton, T. (2023, octubre 14). Response Time Testing. Guru99. Recuperado de https://www.guru99.com/response-time-testing.html

Kinetica. (2023). Precios. Accedido 24 noviembre de 2023 Recuperado de https://www.kinetica.com/pricing/

LoadView. (2020, 4 de noviembre). ¿Cuál es el propósito de una prueba de esfuerzo? Prueba de API, Pruebas de rendimiento. Recuperado 3 diciembre 2023 de https://www.loadview-testing.com/es/blog/cual-es-el-proposito-de-una-prueba-de-esfuerzo/

National Cyber Security Centre. (2023, 30 de marzo). Cyber Security Toolkit for Boards. Recuperado de https://www.ncsc.gov.uk/collection/board-toolkit/embedding-cyber-security-into-your-organisation

OWASP. (2023). OWASP Top 10. Recuperado de https://owasp.org/Top10/

Red Salud. (2022, abril). ¿Por qué hacerse exámenes y chequeos preventivos? Recuperado de https://www.redsalud.cl/salud-y-cuidados/por-que-hacerse-examenes-y-chequeos-preventivos

Sucuri. (2023). Sucuri 2022 Website Threat Research Report. Recuperado de https://sucuri.net/wp-content/uploads/2023/04/Sucuri_2022-Website-Threat-Research-Report.pdf

Tu Página. (2023). Precios. Accedido 24 noviembre de 2023 Recuperado de https://www.tupagina.cl/#precios

Ubisend. (2021). The Chat-bot Statistics Cheatsheet (versión 4). Recuperado 3 diciembre 2023 de https://www.ubisend.com/white-papers/ubisend_chat-bot_statistics_cheatsheet_v4.pdf

W3Techs. (2023). Overview of Web Technologies: Content Management. Recuperado de https://w3techs.com/technologies/overview/content_management

15. Anexo:

15.1. Código de la orden médica PDF:

```
const styles = StyleSheet.create({
  caja: {…
  },
  iframe: {…
  },
  body: {…
  },
  title: {…
  },
  Fecha: {…
  },
  Diag: {…
  },
  Edad: {…
  },
  Nombre: {
    width: "250px",
    margin: 12,
    fontSize: 12,
    textAlign: "left",
    marginTop: -28,
  },
  Rut: {…
  },
  image: {…
  },
  Fondo: {…
  },
  header: {…
  },
  id: {…
  },
  Reali: {…
  },
  examen: {…
  },
  fir: {…
  },
  Dra: {…
  },
  list: {…
  },
  fixing: {…
  },
});
```

Código 3 que define el estilo de cómo se debería ver la orden médica. Elaboración propia.

El código muestra la función StyleSheets que se utilizó para definir como se vería una orden medica solo se muestra uno de los 19 que se utilizaron, ya que los demás son parecidos y no se lograba una buena imagen que capturen todos desplegados. Lo que cabe destacar el problema que se tuvo con la firma del doctor, cuando se llenaba con exámenes la orden médica y sobre pasaba el tamaño de la hoja la firma saltaba a una página en blanco, esto causaba un problema, ya que por términos legales no se puede tener una firma electrónica en una hoja en blanco, se solucionó agregando el Style llamado fixing que lo único que hace es todo lo que lo rodea se comporta como un componente fijo se puede ver el código que genera la orden en el Código 4.

```
189  const LaboratorioDoc = () => (
190    <Document pageLayout="singlePage">
191      <Page size="A4" style={styles.body} debug={false}>
192        <Image style={styles.image} fixed src={doc} />
193
194        <Text style={styles.header} fixed>
195          ORDEN DE EXAMEN
196        </Text>
197        <Text style={styles.Edad} fixed debug={false}>
198          Edad: {UserAge}
199        </Text>
200        <Text style={styles.Rut} fixed debug={false}>
201          Rut: {Rut}
202        </Text>
203        <Text style={styles.Nombre} fixed debug={false}>
204          Nombre: {Name}
205        </Text>
206        <Text style={styles.Fecha} fixed debug={false}>
207          Fecha: {DateMedical.getDate()}/{month}/{DateMedical.getFullYear()}
208        </Text>
209        <Text style={styles.Diag} fixed debug={false}>
210          Diagnostico: Control Médico
211        </Text>
212        <Text style={styles.Reali} fixed>
213          Laboratorio
214        </Text>
215        <View height={10}>
216          <Text style={styles.examen}>{Laboratorio}</Text>
217        </View>
218        <View height={72} style={styles.fixing} fixed>
219          <Image style={styles.fir} fixed src={firm} debug={false} />
220          <Text style={styles.Dra} fixed>
221            Jorge Eduardo Caro Diaz
222          </Text>
223          <Text style={styles.Dra} fixed>
224            Rut: 15.946.308-7
225          </Text>
226        </View>
227      </Page>
228    </Document>
229  );
```

Código 4 genera la orden medica esta fue dividida en 3 PDF distinto solo se muestra uno, ya que los demás son exactamente igual solo cambia que exámenes son colocados. Elaboración propia.

Código de una orden medica específicamente la que entrega los exámenes categorizados como laboratorio también existen la orden medica de Imagenología y Procedimientos, el código es exactamente lo mismo solo cambia la línea 216 específicamente la variable Laboratorio que se coloca Imagenología y Procedimientos respectivamente.

```
316   const LabButton = () => {
317     if (Laboratorio.every((item) => item === undefined)) {
318       return;
319     } else {
320       return (
321         <div>
322           <View style={styles.caja} className="PDF_View">
323             <div className="PrintOrderDiv">
324
325               <PDFDownloadLink
326                 document={<LaboratorioDoc />}
327                 fileName="Laboratorio">
328                 {({ loading }) =>
329                   loading ? (
330                     <div
331                     className="PrintOrderButton ButtonLab text-center rounded-pill">
332                     Cargando
333                   </div>
334                   ) : (
335                     <div
336                     className="PrintOrderButton ButtonLab text-center rounded-pill">
337                       Descargar orden Laboratorio
338                    </div>
339                     )
340                   }
341               </PDFDownloadLink>
342
343
344             </div>
345           </View>
346         </div>
347       );
348     }
349   };
350
```

Código 5 Botones de descarga de la orden médica. Elaboración propia.

Código de los botones de descarga, se mostró solo el código de Laboratorio, dado que para los otros exámenes son parecidos solo cambia en la línea 317, ya que esta revisa si la variable que contiene los exámenes específicamente de laboratorio si esta se encuentra vacía el botón se esconde, el cambio que se hace para los demás es remplazar la variable con Imagenología y Procedimientos. El botón también cambia de cargando a descargar orden, para mostrar cuando el botón se puede apretar y que descargue la orden médica.

15.2. Evaluación económica fuentes de los costos

- Presupuesto: Premio Avante3: 45 millones (fuente: https://www.infodefensa.com/texto-diario/mostrar/4305985/equipos-controlo-amanda-ganan-desafio-avante-3-armada-chile)
- Precio de desarrolladores: 500.000 líquido, 573.000 bruto, multiplicado por 3 desarrolladores + IVA.

Licencias:

- Microsoft 365: 53.990 CLP al año por miembro (fuente: https://www.microsoft.com/es-cl/microsoft-365/buy/compare-all-microsoft-365-products)
- NordPass: 44,38 dólares al año por miembro. (fuente: https://nordpass.com/es/plans/)
- Antivirus: 49,99 dólares al año por miembro. (fuente: https://offer.intego.com/es/mac-av?aff_id=5774&coupon=1Y19X2&aff_sub4=1Y29X2&vpn=1&mo=1&source=2874953)
- Jira: 650 dólares al año por 3 integrantes (fuente: https://www.atlassian.com/software/jira/service-management/pricing)
- Miro: 8 dólares al mes por miembro. (fuente: https://miro.com/es/pricing/)
- Arquitectura cloud mensual diagrama de arquitectura (https://azure.microsoft.com/en-us/pricing/calculator/)
- Base de datos: 12,48 dolares
- Web app para la API: 13,14 dolares
- Firewall: 912,50 dolares. (Fuente: https://learn.microsoft.com/en-us/azure/firewall/overview)
- Static web app para la página: 213,78 dolares.
- Dominio: 3,20 dolares
- Desarrollador senior part time: 18 millones anual
- Leader proyecto: 9millones y 600mil CLP anual

Costos variables

- Flow: 3,19% + iva de la compra. (Fuente: https://sandbox.flow.cl/tarifas.php)
- Liorren: 0,0002 uf por boleta (Fuente: https://www.lioren.cl/docs#/api-emisiondte)

15.3. Código de la llamada a la API

```
104   const pay = () => {
105     setPaying(true);
106     var params = {
107       commerceOrder: Math.floor(Math.random() * (2000 - 1100 + 1)) + 1100,
108       subject: "Pago de orden medica",
109       currency: "CLP",
110       amount: SubTotal,
111       email: Email,
112       urlConfirmation: "https://flowapiymc.azurewebsites.net/apiFlow/payment_confirm",
113       urlReturn: "https://flowapiymc.azurewebsites.net/OrderConfirmedRoute"
114     };
115
116     var requestOptions = {
117       method: 'POST',
118       redirect: 'follow',
119       headers: {
120         "Content-Type": "application/json",
121       },
122       body: JSON.stringify(params)
123     };
124     fetch("https://flowapiymc.azurewebsites.net/apiFlow/create_order", requestOptions)
125       .then(response => response.text())
126       .then(result =>
127         window.location.replace(result.slice(13, -2)))
128       .catch(error => console.log('error', error));
129   };
```

Código 6 muestra la función que manda la pregunta a la API con los datos de la página. Elaboración propia.

Código de la función pago que genera la llamada a la API con los datos del usuario y el monto a pagar, también se envían las direcciones de confirmación y retorno para que la API sepa donde tiene que mandar la información en el caso de que el pago fue realizado se envía a la dirección de confirmación para después enviar la boleta sino retorna al usuario a la página de pago.

15.4. Código que envía la boleta

```
app.post('/OrderConfirmedRoute', async(req, res) => {
  try{
    const secret = await client.getSecret("Bear");
     let params = {
       token: req.body.token
     };
     let serviceName = "payment/getStatus";
     const flowApi = new FlowApi(config);
     let response = await flowApi.send(serviceName, params, "GET");
     //Actualiza los datos en su sistema
     if (response.status == 2){
      const gmailboleta = response.payer;
      var raw = JSON.stringify({…
       });
       var requestOptions = {
         method: 'POST',
         redirect: 'follow',
         headers: {
           "Content-Type": "application/json",
           "Authorization": secret,
         },
         body: raw
       };
       fetch("https://www.lioren.cl/api/boletas", requestOptions)
         .then(response =>  response.json())
         .then(result=>send_mail(gmailboleta,result.pdf))
         .catch(error => console.log('error', error));
       res.redirect('https://yomecontrolo.cl/OrderConfirmedRoute');
     }else{
       res.redirect('https://yomecontrolo.cl/ExamRecommendationRoute');
     }
  }catch (err) {
   console.error(err.message);
   res.json({ error: err });
 }
  });

```

Código 7 Función post que se activa después de recibir la confirmación de un pago realizad. Elaboración propia

Código de la API que espera la confirmación de Flow y revisa el estado del pago si es aprobado este se envía a la dirección “https://yomecontrolo.cl/OrderConfirmedRoute” donde se encuentran los botones de las distintas ordenes medicas vistos en el Código 5, si no es aprobado este es devuelto a la página donde selecciona que exámenes quieren pagar y tendrán que volver a pasar por la pasarela de pago.

15.5. Botones de pago y redirigir

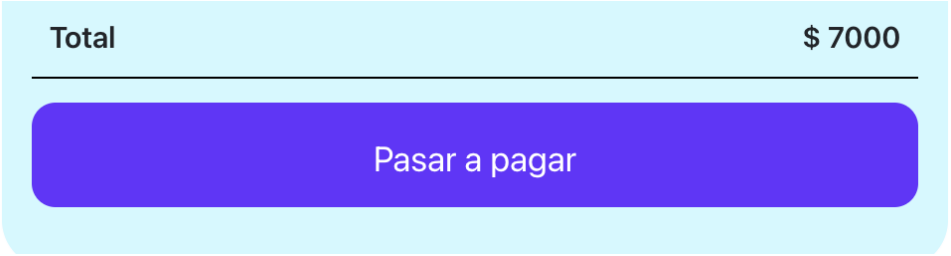

Imagen 7 botón de pasar a pagar

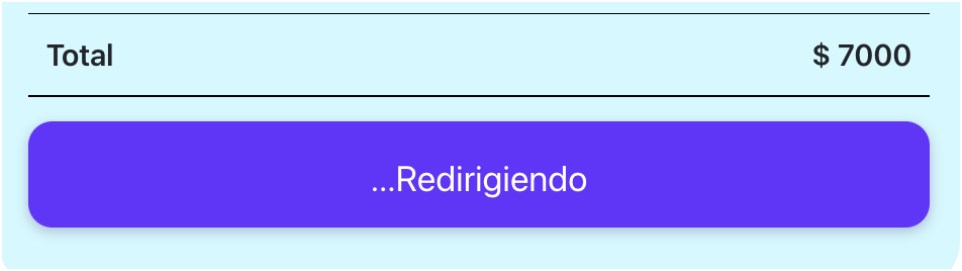

Imagen 8 después de presionar el botón pasar a pagar el nuevo botón que aparece

En la imagen 7 es el botón antes de ser presionado al momento de apretar esta cambia a la imagen 8 que es el mismo botón, pero muestra Redirigiendo y se desactiva para que el usuario no genere más llamadas a la API, por detrás se ejecuta la función revisada en el anexo 15.3.