Instituto Politécnico Nacional

Escuela Superior de Ingeniería Mecanica y Electrica
Unidad Culhuacan
Ingeniería en Computación

REPORTE TÉCNICO

Desarrollo de una plataforma para la ejecución de pruebas de carga a partir de la definición de un script con JavaScript

---

Presenta
Jonathan Eduardo García García jgarciag1404@alumno.ipn.mx

Asesor del proyecto:

M. en I. TIRSO MARTÍNEZ REYES

27 de diciembre de 2023

Índice

1. Resumen

Esta plataforma permite la ejecución de flujos de negocio específicos para escenarios no estáticos de pruebas de carga utilizando como lenguaje principal Javascript con el cual se desarrollaron las principales funciones de la aplicación y Amazon Web Services para el almacenamiento y aprovisonamiento está, incluyendo el desarrollo un servicio API para la comunicación entre las peticiones y la base de datos.

Palabras Clave: Pruebas de carga, Javascript, endopoint, API, multi-factor, asincrónico.

2. Introducción

Las pruebas de carga para sistemas de información basados en la web se llevan a cabo de manera limitada. Tradicionalmente, estas pruebas siguen un conjunto preestablecido de peticiones dirigidas a servicios web específicos, donde los scripts de prueba dictan el flujo de ejecución. Este enfoque no permite adaptar o incluir lógicas adicionales que podrían ser necesarias para ciertos flujos de negocio específicos. Como resultado, hay ciertas restricciones en las pruebas de carga, especialmente aquellas que involucran validaciones complejas, procesamiento asincrónico y sistemas de autenticación multifactor, que no pueden ser adecuadamente evaluadas dentro del esquema de prueba convencional.

3. Planteamiento del Problema

Actualmente el diseño y ejecución de pruebas de carga para sistemas de información Web se basan en un flujo pre-establecido de peticiones a un conjunto de servicios Web (endpoints) sincronizados por scripts de pruebas previamente definidos, lo cual no deja espacio para lógica intermedia necesaria para flujos de negocio particular y por ende, limita el conjunto de flujos de ejecución para la realización de pruebas que requieren validaciones, procesamiento asíncrono y esquemas de autenticación multi-factor.

4. Objetivos

4.1. Objetivo General

Implementar una plataforma que permita ejecutar scripts de pruebas de carga diseñados en JavaScript aprovechando la multitud de paquetes opensource disponibles en el registro de código (npm) y con ello extender las capacidades de las pruebas de carga a escenarios mucho más complejos, recopilando los resultados y métricas en un entorno distribuido, centralizando los resultados de la instrumentación en el tablero de control.
Desarrollar una interfaz (tablero de control) donde el usuario pueda ejecutar las pruebas de carga que haya diseñado y visualice las métricas de ejecución e información relevante.

4.2. Objetivos específicos

1. Desarrollar un servicio web que cargue la definición de la prueba de carga a partir del código JavaScript.
2. Desarrollar un servicio web que permita gestionar y configurar las instancias de cada prueba de carga durante su ejecución.
3. Diseñar un servicio web que recupere y almacene las métricas de ejecución para cada prueba de carga.
4. Diseñar una base de datos que permita almacenar la información y métricas relevantes a cada ejecución.
5. Diseñar y programar una interfaz gráfica (tablero de control) que presente los resultados de la ejecución y la información recopilada de forma simplificada y unifique los resultados de cada uno de los ejecutores.

5. Límites y alcances

1. **Alcances.**
   a) La plataforma soporta JavaScript como lenguaje de definición para las pruebas de carga.
   b) Adecuado para testers con conocimiento particular del flujo de la lógica de negocio
   c) Adecuado para testers con conocimiento del lenguaje JavaScript
2. **Límites.**
   a) Diseñado para utilizar los servicios de AWS como aprovisionamiento
   b) El tamaño de las pruebas está limitado por la configuración especifica de la cuenta de AWS que se esté utilizando

6. Justificación

Hoy en día con el creciente aumento de necesidades de la industria en materia de ciberseguridad y pruebas de caja blanca, se requieren diseñar y ejecutar pruebas de carga de flujos de ejecución los cuales requieren integrar componentes de diversa naturaleza y origen. Quienes requieren de estas capacidades en materia de pruebas de carga optan por realizar scripts, personalizados lo cual implica una importante inversión de tiempo y esfuerzo, así como una limitación en la cantidad, calidad y profundidad de las métricas para evaluar el desempeño del sistema.
Se busca resolver con esta plataforma la falta de esta automatización y conseguir estandarizar el diseño de estas pruebas de carga.

7. Estado del Arte

[1] Loader.io es un servicio de pruebas de carga gratuito que le permite realizar pruebas de esfuerzo en aplicaciones web y APIs con conexiones simultáneas.

[2] Herramienta de pruebas de carga de código abierto con conexiones simultáneas e implementación Cloud y local.

[3] Permite realizar pruebas de esfuerzo a sitios web, aplicaciones web y APIs con conexiones simultáneas desde la nube

[4] Herramienta de pruebas de UI con paralelización en navegadores reales.

8. Marco Teórico

8.1. Sistemas de Información

[5] Los sistemas de información constituyen uno de los aspectos estratégicos claves para el buen hacer de la empresa. Para ello es necesario que la totalidad de la organización esté concienciada de su utilidad, tanto por parte de la alta dirección, la cual ha de tenerlos en cuenta a la hora de realizar el proceso de planificación estratégica de la empresa, como por parte de los distintos usuarios de la empresa. Ha de existir una política de información y motivación dentro de la empresa. Si esto se lleva a cabo, la empresa logrará superar a sus competidores, podrá aumentar su poder de negociación e incluso podrá evitar la entrada de nuevos competidores logrando la denominada “ventaja competitiva sostenible”.

8.2. Pruebas de carga

Una prueba de carga para un sistema web es un proceso que evalúa cómo el sistema se comporta al someterlo a un volumen de trabajo que replicaría un uso normal o pico de recursos por parte de los usuarios finales. Esto implica simular múltiples usuarios accediendo simultáneamente a diferentes partes del sistema web y realizando diversas operaciones para medir su rendimiento y estabilidad.
El objetivo principal de las pruebas de carga es identificar y diagnosticar cuellos de botella, problemas de rendimiento, y asegurarse de que el sistema puede manejar la cantidad esperada de tráfico sin deteriorarse ni fallar. Esto puede incluir comprobar la velocidad de respuesta de las páginas, la eficiencia del servidor y la base de datos bajo carga, la gestión de recursos del sistema, y la escalabilidad general de la aplicación web.
La prueba de carga difiere de otros tipos de pruebas, como las pruebas de estrés o pruebas de resistencia, en que no busca romper el sistema o descubrir su punto de fallo. En su lugar, trata de imitar el uso diario para garantizar que el rendimiento del sistema será adecuado cuando esté en funcionamiento y bajo la demanda de los usuarios reales.

8.3. Base de datos

Una base de datos es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Una base de datos es usualmente controlada por un sistema de gestión de base de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones que están asociados con ellos, se conocen como un sistema de base de datos, que a menudo se reducen a solo base de datos. Los datos dentro de los tipos más comunes de bases de datos en funcionamiento hoy en día se modelan típicamente en filas y columnas en una serie de tablas para que el procesamiento y la consulta de datos sean eficientes. Luego se puede acceder, administrar, modificar, actualizar, controlar y organizar fácilmente los datos. La mayoría de las bases de datos utilizan lenguaje de consulta estructurado (SQL) para escribir y consultar datos.

8.4. Tecnologias y software

Amazon Dyanmo	Es una base de datos no-sql utilizada para almacenar la información, metricas y metadatos resultantes de la ejecución y configuración de las pruebas de carga. Debido a la cantidad de información que utiliza este proyecto se requiere un modelo de datos semi-estructurado para almacenar y recuperar información de forma eficiente y ordenada.
Amazon EC2	Amazon Elastic Compute Cloud (Amazon EC2) proporciona capacidad de computación escalable bajo demanda en la nube, se utiliza como aprovisonamiento dinamico donde se ejecuta cada una de ka
Javascript	Es uno de los lenguajes mas utilizados por los desarrolladores, se puede ejecutar en cualquier plataforma debido a que es interpretado y permite intregrar caracteristicas modernas para describir el flujo de las pruebas de carga.
GitHub	Control de versiones, en cada confirmación se especifican los cambios realizados.
Application Programming Interface	Se utiliza como intermediario para la comunicación entre la base de datos alojada en AWS y la aplicación.
Amazon Web Services	Se utiliza para alojar el motor de base de datos mediante RDS , el foro de comentarios y el servicio API

9. Desarrollo

Para el inicio del desarrollo del proyecto se realizo una investigación inicial mediante un cuestionario con el cual se obtuvieron los siguientes datos cuantitativos que nos han dado una mayor viabilidad del proyecto, así como las tendencias de uso de los usuarios los cuales se muestran a continuación:

Figura 1: Medios por los cuales los alumnos consultan su horario. Hemos podido observar que en su mayoría los alumnos acceden al SAES para consultar su horario, dejando otras alternativas en segundo término.

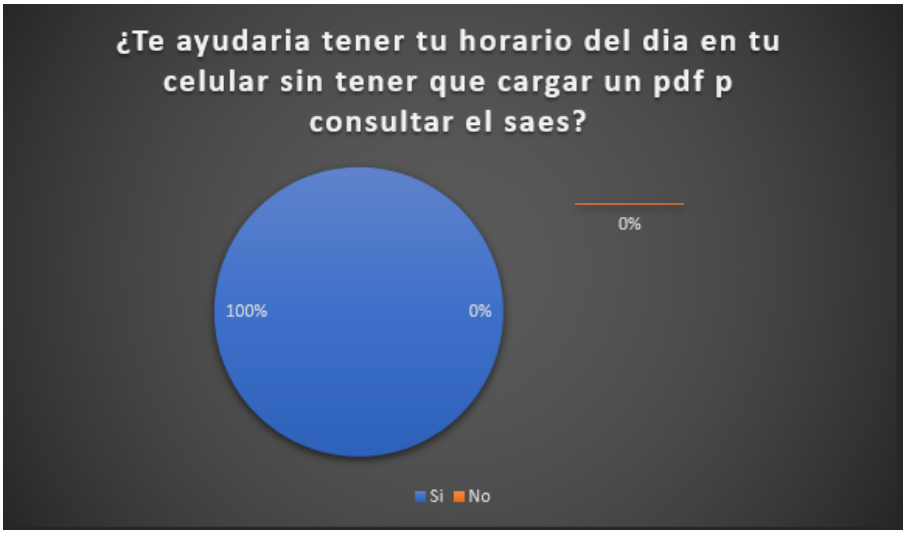


Figura 2: Muestra de alumnos que preferirían tener su horario de manera más eficiente que en un PDF o consultar el saes para obtener esta información.

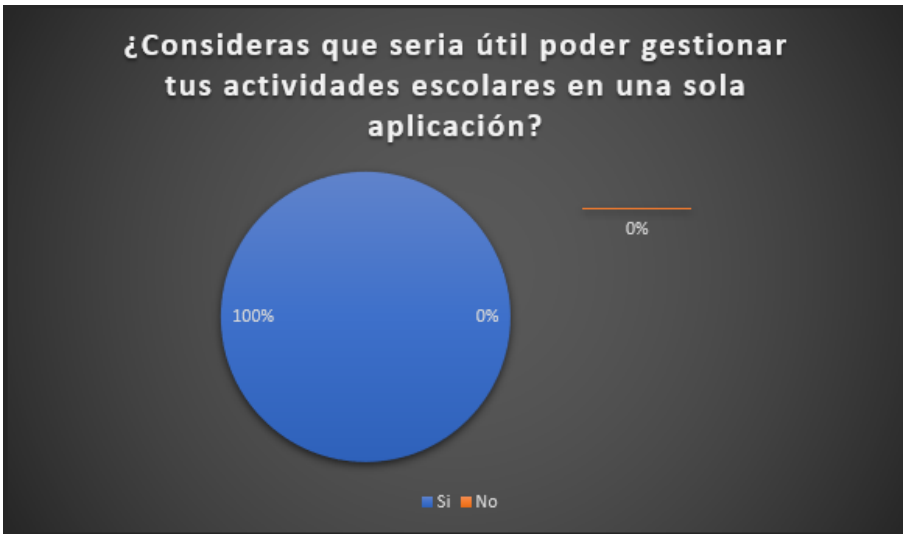


Figura 3: Muestra de alumnos que preferirían gestionar en una sola aplicación todas sus actividades.

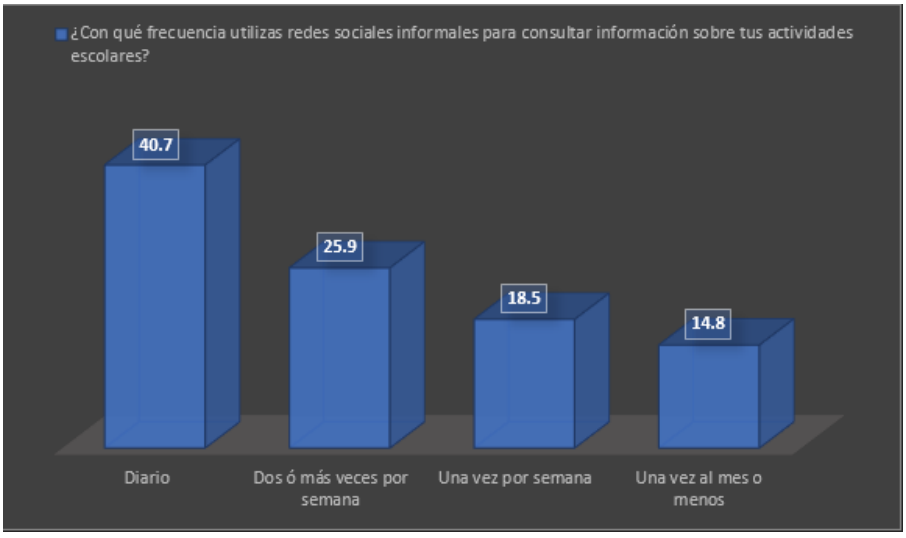


Figura 4: Muestra de las veces que los alumnos visitan diversas redes sociales para obtener información de sus actividades escolares.

Figura 5: Muestra de alumnos que sí utilizan herramientas para llevar a cabo la organización de sus actividades.

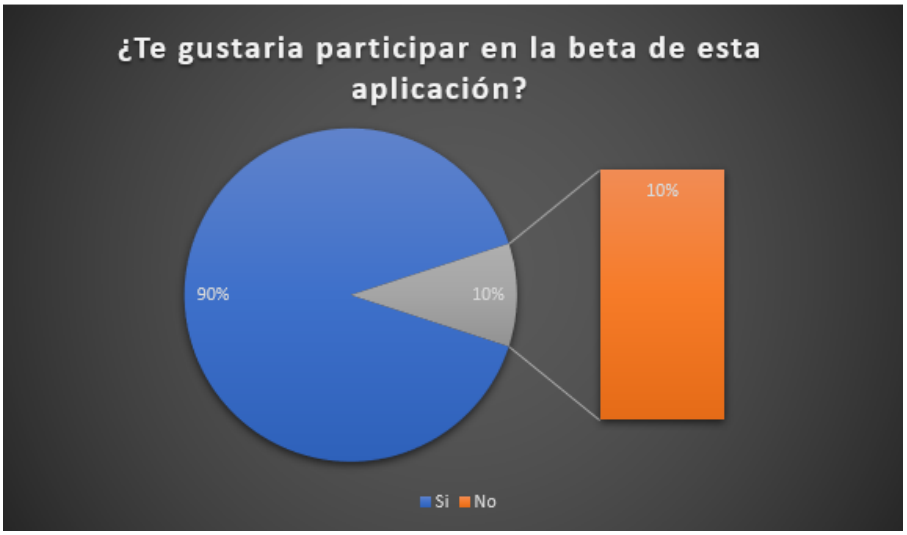


Figura 6: Versión beta del Sistema. Hemos notado un interés notorio de los alumnos para probar la versión beta de la aplicación.

Figura 7: Versiones de sistema. En esta muestra se ha observado que el 96.3 % corresponde a usuarios del Sistema Operativo Android y el 3.7 % a usuarios que utilizan IOS.

9.1. Cronograma de actividades

Una parte relevante del proyecto ha sido la bitácora de actividades, en ella se ven plasmados los requerimientos iniciales así como los ajustes realizados durante el desarrollo del proyecto, la bitácora nos ayudó a tener un control más adecuado de las actividades a desarrollar. Los requermientos se enlistan con la fecha en la cual se dió inicio.

Requerimiento	**Fecha de inicio**
Diseños propuestos para Login	Fecha: 31/1/2021 19:30:00
Implementación de FingerPrint	Fecha: 18/2/2021 18:43:45
Importar el horario del saes leyendo html una vez que el usuario haya iniciado sesión	Fecha: 28/2/2021 21:00:26
Pantalla de ajustes	Fecha: 28/2/2021 21:01:50
Pantalla para ver las tareas mostrando los días en los que hay actividades, al final mostrar que no hay más tareas	Fecha: 2/6/2021 0:00:00
Pruebsa en el acomodo de imágenes para dar de alta tareas	Fecha: 28/2/2021 23:03:05
Color distinto por matera	Fecha: 8/3/2021 21:54:18
Concluir ventana alta de tareas	Fecha: 15/3/2021 19:02:53
Vista de horario como tabla	Fecha: 15/3/2021 19:24:27
Recopilar nombre de las escuelas y enlaces de acceso al saes	Fecha: 21/3/2021 21:52:35
Obtener la información del perfil según el mockup desde el saes	Fecha: 15/3/2021 19:25:09
Pantalla para editar y dar de alta tareas	Fecha: 21/3/2021 23:24:42
Vista de detalle por tarea	Fecha: 21/3/2021 23:25:06
Dar de alta recordatorios de exámenes ,otros eventos	Fecha: 21/3/2021 23:29:38
Pantalla de perfil del estudiante	Fecha: 21/3/2021 23:31:08
Exportar horario en PDF	Fecha: 21/3/2021 23:43:53

Aplicación móvil		
Plataforma	Android	iOS
Versión mínima	API 23+ 6.0.1 Marshmallow	iOS 9
Versión destino	API 30 Honeycomb	iOS 14.5
Procesador	arm64v8a, armeabiv7a, x86_64	A9, A10, A11, A12
Hardware	Dispositivos móviles y tabletas Android	Dispositivos móviles iOS

Componente	**Origen**
Conectividad a internet	Requerido
Cámara	Deseable opcional

Proveedor de base de datos				
Version SQL	**CPU**	**RAM**	**Almacenamiento**	**Tipo de almacenamiento**
Sqlserver express 14.00.3356.20.v1	1	1Gb	20 Gbs	SSD
Proveedor de servicios web				
Entorno	**CPU**	**RAM**	**Almacenamiento**	**Tipo de almacenamiento**
.NET Core 3.1 (C#/PowerShell)	1	512 Mb	50 Gbs	SSD

9.2. Implementacion de ISO 2000 en el proyecto

ISO 2000 En base a la ISO 2000 se deben desarrollar los siguientes pasos:

- Evaluar necesidades y metas de organización:
  Se evaluaron las necesidades de los alumnos de la carrera de computación y la necesidad más grande fue la falta de organización en las tareas cotidianas escolares y la meta a desarrollo es una aplicación para poder administrar sus actividades escolares.
- Obtener información:
  Se realizaron encuesta a los alumnos de la carrera de computación para saber el uso cotidiano del celular y su forma de organizar sus actividades escolares.
- Nombrar consultor:
  Profesor Oscar Cruz García quien nos ha ayudado en la estructuración de las partes del proyecto.
- Toma de conciencia y formación:
  a) La protección de los datos sensibles de los usuarios
  b) Los objetivos de calidad pertinentes para el proyecto
  c) La contribución que haremos a los alumnos y beneficios que mejoraran el desempeño de ellos.
  d) Y también lo que implica incumplir los requisitos del Sistema proyecto.
- Análisis de brech:
  Evaluamos las diferencias entre el desempeño real que tendrá el proyecto y el desempeño esperado que teníamos al inicio de este.
- Revisión o definición de procesos:
  ¨ Se hizo un análisis de los requerimientos.
  ¨ Se diagnosticaron los requerimientos funcionales y no funcionales.
  ¨ Formación de las etapas de implementación de los requerimientos.
  ¨ Verificación de los estándares de calidad que debe tener le proyecto.
- Suministrar personal:
  Se organizaron las tares entre los integrantes del equipo para así tener un mayor desempeño en el desarrollo.
  Establecer cronograma:
  Al ya tener definidos los alcances del proyecto se implementó el cronograma de actividades utilizando Trello para llevar una organización de ellas.

9.3. Diagrama de Casos de Uso

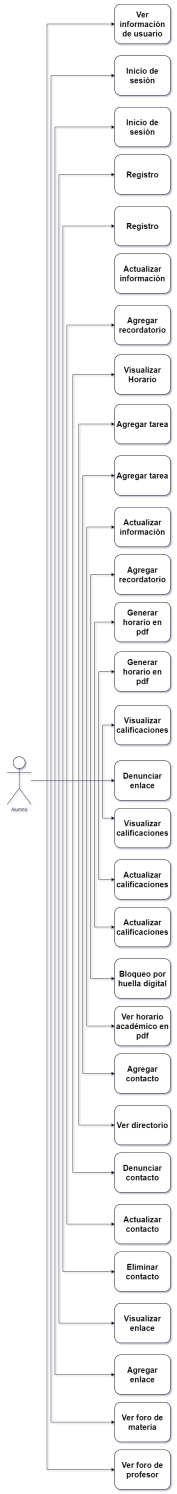


Figura 8: Diagrama de casis de uso entre la aplicación y el alumno.

9.4. Matriz de trazabilidad de casos de uso.

	Actor: Alumno
CUS1: Ver información de usuario	x
CUS2: Inicio de sesión	
CUS3: Registro	x
CUS4: Actualizar información	
CUS5: Agregar tarea	x
CUS6: Agregar recordatorio	x
CUS7: Visualizar horario	x
CUS8: Generar horario en PDF	x
CUS9: Visualizar calificaciones	x
CUS10: Actualizar calificaciones	
CUS11: Activar / desactivar bloqueo por huella digital	x
CUS12: Activar / desactivar barra lateral de horario	x
CUS13: Ver horario académico en PDF	x
CUS14: Ver directorio	x
CUS15: Agregar contacto	x
CUS16: Denunciar contacto	x
CUS17: Actualizar contacto	x
CUS18: Eliminar contacto	x
CUS19: Visualizar enlace	
CUS20: Agregar enlace	x
CUS21: Denunciar enlace	x
CUS22: Ver foro de comunidad	x
CUS23: Ver foro de profesor	x
CUS24: Ver calendario escolar	x
CUS25: Mantener expandidas las tareas	x

En donde:

CUS: Hace referencia al caso de uso
Usuario: El actor dentro de esta matríz de trazabilidad.

9.5. Matriz de trazabilidad

ID	REQUISITO	TIPO	PRIO	ESTADO	OBJETIVO	ENTREGABLE	ESTADO	VALIDACIÓN
1	Protección biométrica	Android/ IOS	Alta	Finalizado	Uso del sistema biométrico por huella para protección de la aplicación.	Desbloqueo de la aplicación con biometría	Testing	17 de ago. a las 20:05
2	Visualizar horario	Saes	Alta	Finalizado	Visualización de horario actual ordenado por clase y colores	Horario Obtenido del saes	Completado	28 de mar. a las 22:59
3	Exportar horario en PDF	Research	Media	Finalizado	Generar el horario en formato PDF	Se obtiene una versión en PDF del horario	Completado	6 de abr. a las 20:28
4	Pantalla de perfil del estudiante	Diseño	Alta	Finalizado	Al ingresar por primera vez se podrá visualizar la información del alumno como carrera, escuela y avance escolar	Venta perfil de estudiante	Completado	17 de ago. a las 20:04
5	Widget de horario	Android	Alta	Finalizado	Desarrollo de widget de horario orientado a Android	Widget de horario [Android]	Completado	08 de jun. a las 00:00
6	Esquema de servidor	API	Alta	Finalizado	Una iniciativa para eliminar el procesamiento en el dispositivo de cliente y apostar por una arquitectura más segura	API	Completado	01 de jul. a las 00:00
7	Crea recordatorios	Page	Alta	Finalizado	Creación de recordatorios por materia con título, descripción, fecha de inicio y fin	Lamba AWS Cloud clusters	Completado	17 de ago. a las 20:04
8	Crea tareas	Page	Alta	Finalizado	Creación de tareas por materia con título, descripción, fecha de inicio y fin, así como adjuntar imágenes	Ventana de altas	Testing	17 de ago. a las 20:05

9	Consulta calificaciones	Saes	Alta	Finalizado	Visualizar las calificaciones y actualizar las mismas	Ventana de calificaciones	Testing	17 de ago. a las 20:05
10	Widget de tareas [Android]	Android	Alta	Finalizado	Desarrollo de widget de tareas orientado a Android	Widget de tareas [Android]	Testing	17 de ago. a las 20:05
11	Compartir Tarea	API/ Diseño	Alta	Finalizado	Se pueden adjuntar enlaces de interés para un grupo de clase	Compartir enlaces	Testing	17 de ago. a las 20:05
13	Notificaciones 10 minutos antes de clase	Research	Alta	Finalizado	Función para compartir las tareas	Compartir tarea	Testing	17 de ago. a las 20:05
12	Compartir enlaces	API/ Diseño	Alta	Finalizado	Recibe notificaciones 10 minutos previos al inicio de cada clase	Recordatorios 10 minutos	Testing	17 de ago. a las 20:05
14	Notificaciones de tarea	Research	Alta	Finalizado	Recibe notificaciones de tareas próximas a vencer	Recordatorios por tarea	Testing	17 de ago. a las 20:05
15	Adjuntar imágenes en tareas	Page	Alta	Finalizado	Al crear una tarea se pueden adjuntar imágenes desde la galería o cámara fotográfica	Actualización ventana de altas	Testing	17 de ago. a las 20:05
16	Información por materia	Page	Alta	Finalizado	Se desplegará información relevante sobre la materia como el grupo y la clase a tomar, así como un foro para estudiantes para discutir temas de la clase así cómo saber qué alumnos están inscritos en ella (compañeros de clase)	Actualización ventana de altas	Testing	17 de ago. a las 20:05

17	Muro de tareas y recordatorios	Page	Alta	Finalizado	La pantalla principal muestra las tareas y recordatorios ordenados por orden de importancia (fecha a expirar)	Actualización ventana de altas	Testing	17 de ago. a las 20:05

9.6. Diagrama de funcionamiento

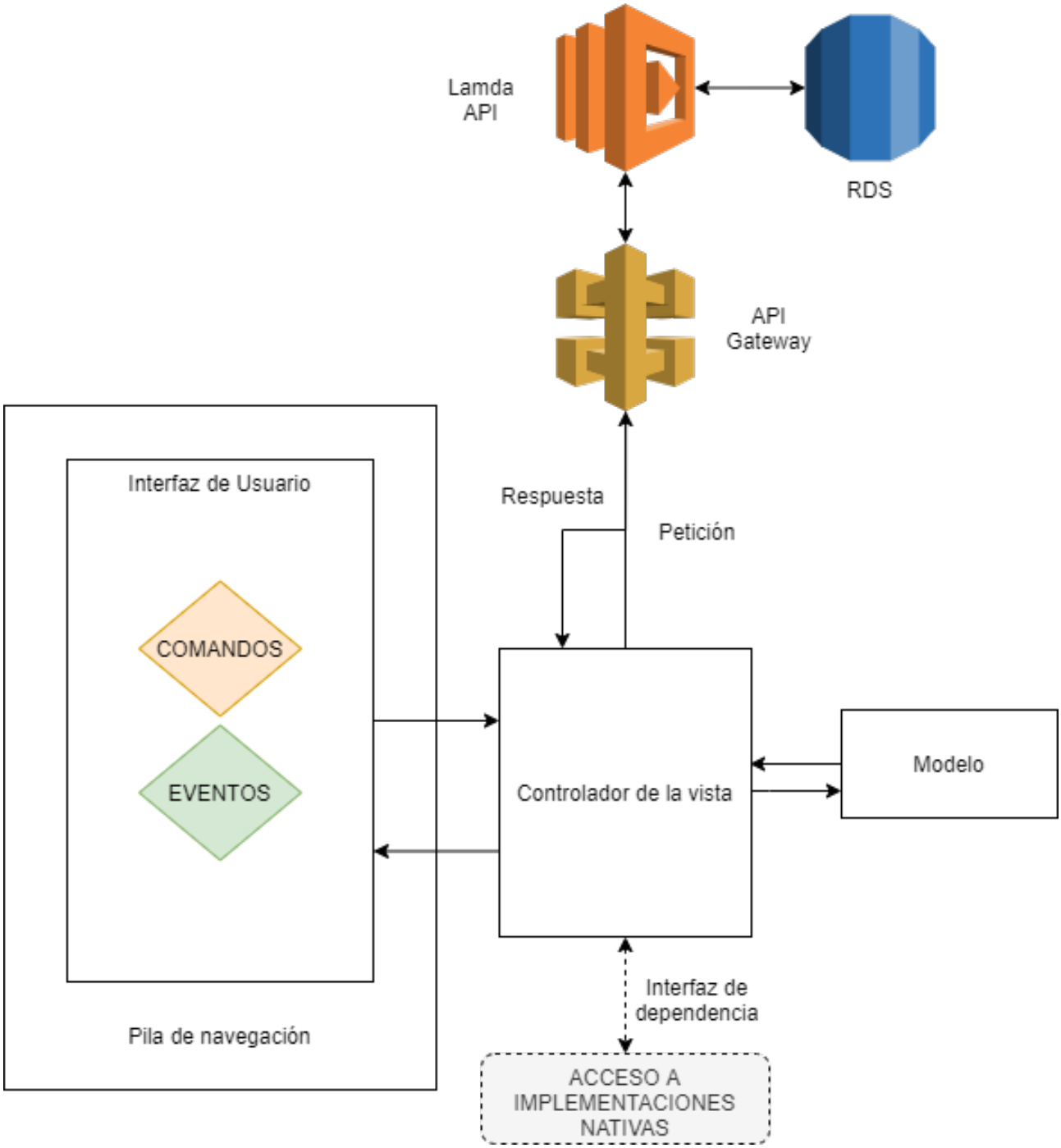


Figura 9: Funcionamiento general de la aplicación y su comunicación con la nube.

9.7. Modelo de solución de software

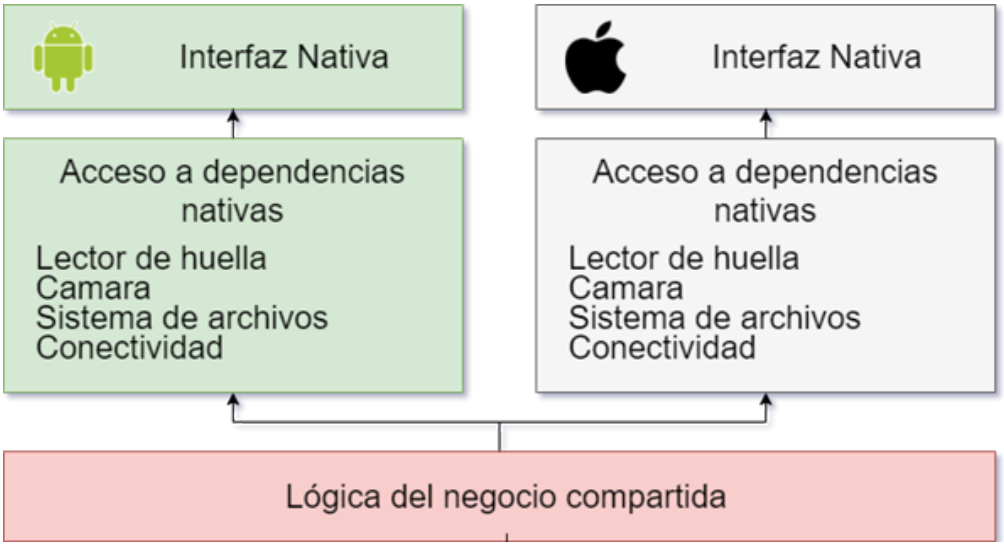


Figura 10: Modelo de arquitectura interna de la solución multiplataforma.

9.8. Diagrama de funcionamiento ventana principal

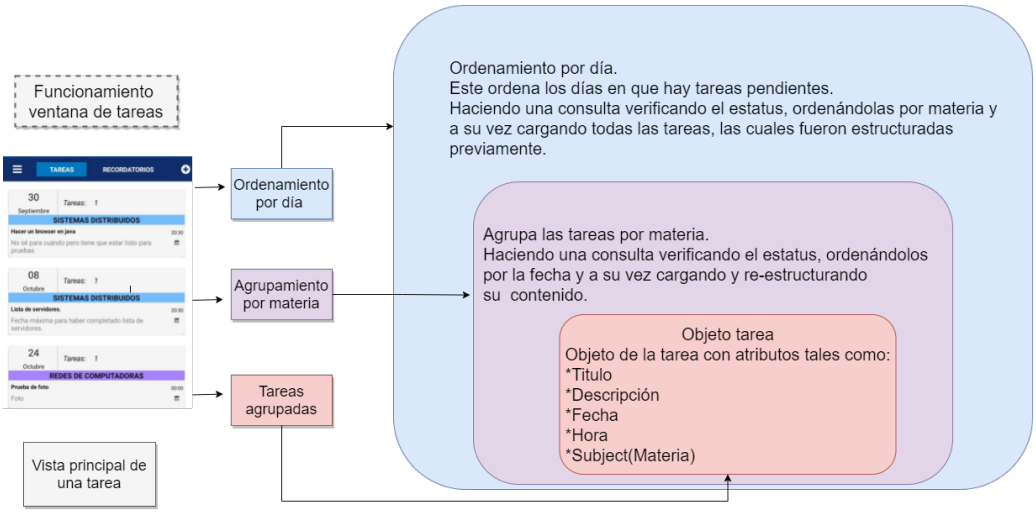


Figura 11: Esquema de la estructura de la ventana de tareas

9.9. Diagrama entidad-relación

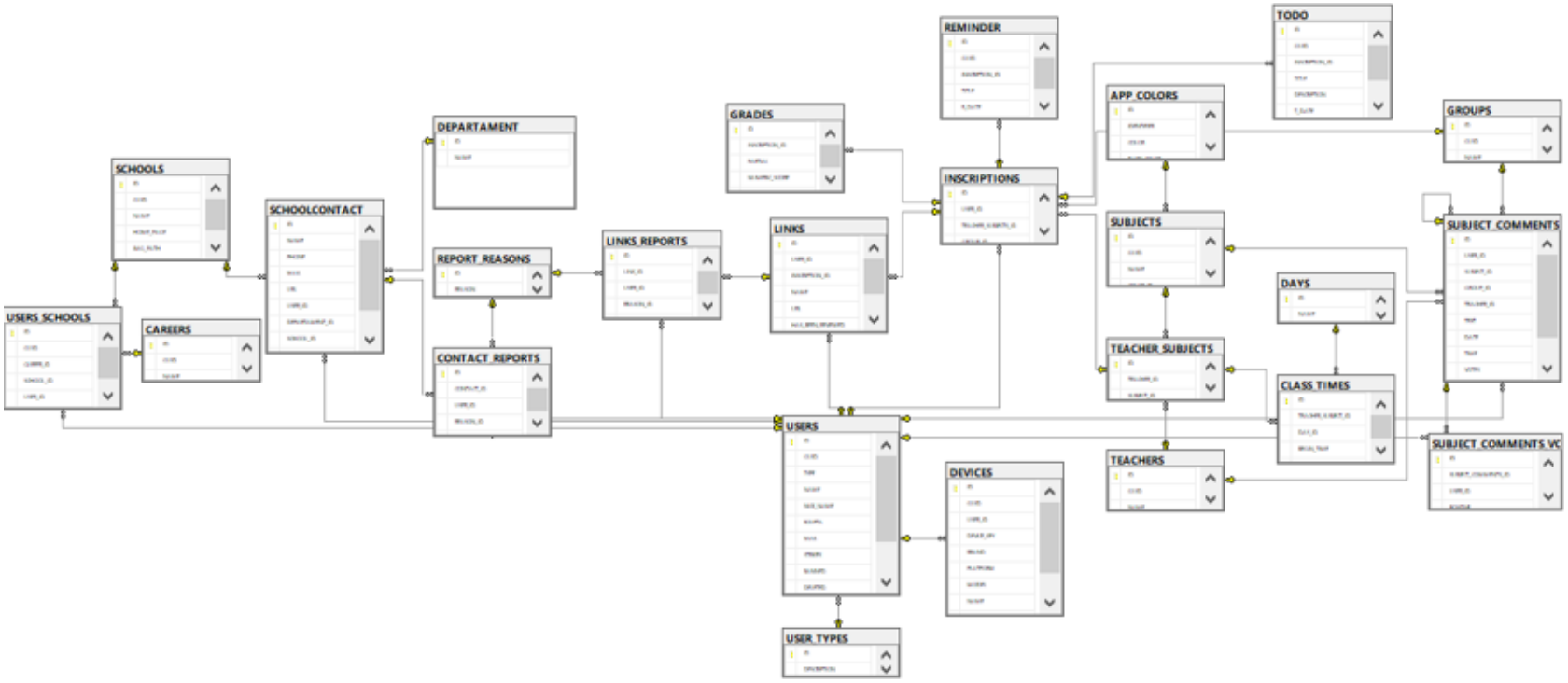


Figura 12: Diagrama entidad-relación de la base de datos alojada en el servicio web.

9.10. Desarrollo de API

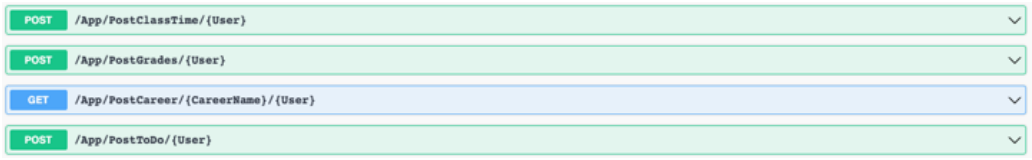


Figura 13: Ejemplos de verbos modificadores y recuperadores de el API

Figura 14: Ejemplo de respuesta del verbo modificador "ShareTodo"

9.11. Llamada desde la aplicación al API

```
public static async Task<Response> PostToDo(TodoBase todo)
{
    await Task.Yield();
    if (todo is null || string.IsNullOrEmpty(todo.Title)
                     || Guid.Empty == todo.Guid
                     || todo.Subject is null
                     || todo.Subject.Id <= 0
                     || todo.Subject.IdTeacher <= 0)
    {
        return Response.Error;
    }
    string json_todo = todo.JsonSerializeObject<TodoBase>();
    WebService WebService = new WebService(Url);
    Kit.Services.Web.ResponseResult result = await WebService.PostAsBody(
        System.Text.Encoding.UTF8.GetBytes(json_todo),
        "PostToDo", AppData.Instance.User.Boleta);
    if (result.Response == "ERROR" || string.IsNullOrEmpty(result.Response))
    {
        return new Response(APIResponseResult.INTERNAL_ERROR, result.Extra);
    }
    return JsonConvert.DeserializeObject<Response>(result.Response);
}
```

Figura 15: Fragmento de código de la llamada desde la aplicación a el servicio Web

```
public async Task<Kit.Services.Web.ResponseResult> PostAsBody(byte[] byteArray, string metho
{
    Kit.Services.Web.ResponseResult result = new Kit.Services.Web.ResponseResult
    {
        HttpStatusCode = HttpStatusCode.Unused
    };
    string geturl = String.Empty;
    try
    {
        geturl = BuildUrl(method, query, parameters);
        var body = new ByteArrayContent(byteArray);
        body.Headers.ContentType = MediaTypeHeaderValue.Parse("application/octet-stream");
        HttpResponseMessage message = await HttpClient.PostAsync(geturl, body);
        result.HttpStatusCode = message.StatusCode;
        result.Response = await message.Content.ReadAsStringAsync();
        return result;
    }
    catch (Exception ex)
    {
        Log.Logger.Error(ex, $"GET: {geturl}");
        result.Response = "ERROR";
        return result;
    }
}
```

Figura 16: Fragmento de código de las llamada POST

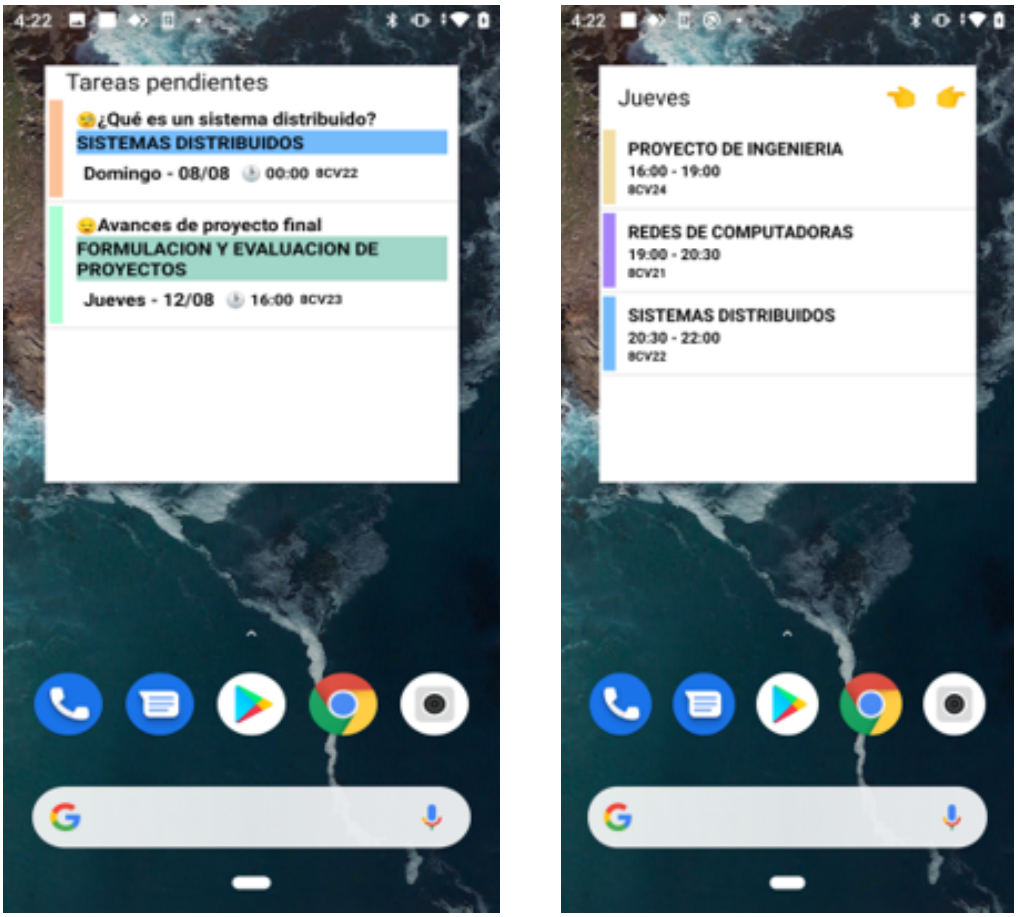
9.12. Fragmento de código de la pantalla para dar de alta una tarea

Figura 17: Fragmento de código del maquetado de la interfaz gráfica en la pantalla para dar de alta una tarea

9.13. Implementación widget nativo

```
android:id="@+id/widget_todos"
style="@style/MainTheme"
android:background="@color/colorOnBackground"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:elevation="4dp" android:padding="0dp"
android:layout_margin="@dimen/widget_margin">
<TextView android:layout_marginLeft="15dp"
                  android:textColor="@color/colorTextPrimary"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:layout_column="0"
                  android:layout_columnSpan="0"
                  android:text="Tareas pendientes"
                  android:textSize="18dp" />
<ListView xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/stack_view_todos"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:gravity="center"
          android:clickable="true"
          android:loopViews="true"
          android:orientation="vertical" />
  <TextView xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/empty_view_todos"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:textColor="@color/colorTextPrimary"
     android:gravity="center"
     android:textStyle="bold"
     android:text="Cargando..."
     android:textSize="20sp" />
</LinearLayout>
```

Figura 18: Fragmento de código de la interfaz gráfica del widget nativo para Android.



(a) Widget de tareas (b) Widget de horario

Figura 19: Widget nativo (Andorid)

9.14. Principales funcionalidades de la aplicación.

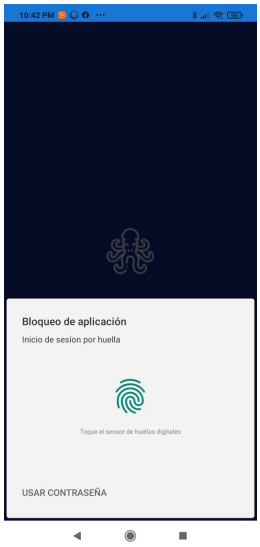


Figura 20: En el inicio de sesión se despliega la opción para ingresar por medio de huella, alternativamente por contraseña.

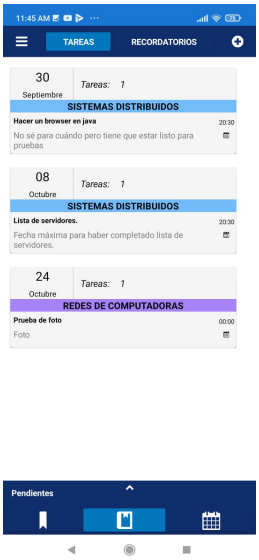


Figura 21: La pantalla de tareas es la primera en mostrarse una vez que se ingresa a la aplicación.

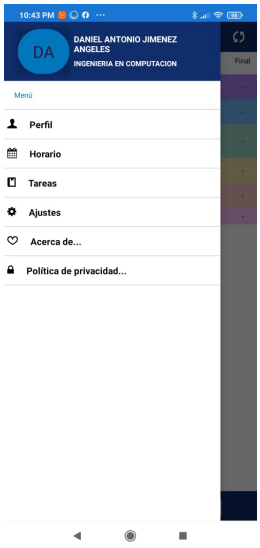


Figura 22: En esta pantalla de muestran los siguientes submenús: Perfil, horario, tareas, ajustes, aderca de y política de privacidad.

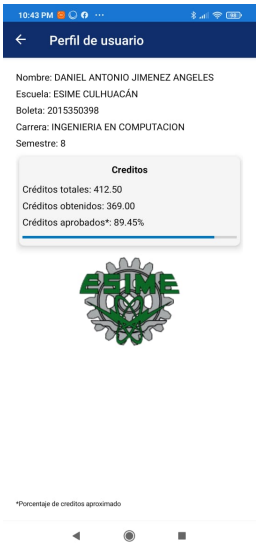


Figura 23: Ingresando a la opción perfil se muestran datos del alumno: Nombre, Escuela, Boleta, Carrea, Semestre, Créditos totales, Créditos obtenidos, Créditos aprobados*.

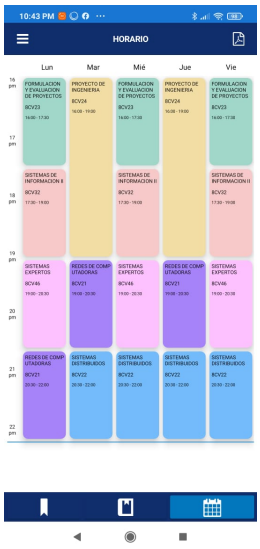


Figura 24: En el inicio de sesión se despliega la opción para ingresar por medio de huella, alternativamente por contraseña.

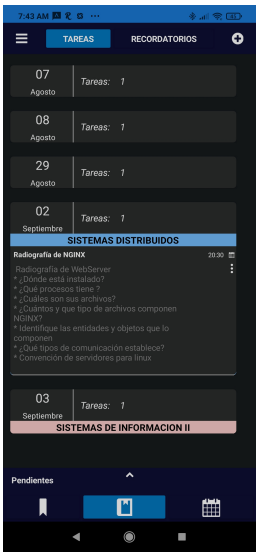


Figura 25: La pantalla de tareas es la primera en mostrarse una vez que se ingresa a la aplicación.

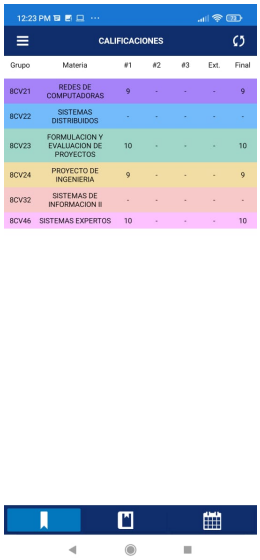


Figura 26: En esta pantalla de muestran los siguientes submenús: Perfil, horario, tareas, ajustes, aderca de y política de privacidad.

Figura 27: Ingresando a la opción perfil se muestran datos del alumno: Nombre, Escuela, Boleta, Carrea, Semestre, Créditos totales, Créditos obtenidos, Créditos aprobados*.

Figura 28: Una vez resuelto el captcha las calificaciones se actualizan en automático.

Figura 29: La pantalla de horario muestra las clases inscritas con dos ejes de referencia; los días de la semana de lunes a viernes y un rango de horas desde el inicio hasta el fin de clases en el día.

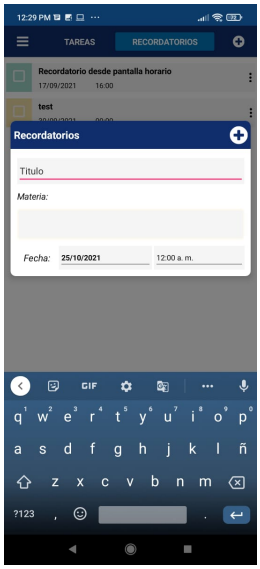


Figura 30: En la opción recordatorio tenemos cuatro campos para editar: Título, Materia, Fecha de entrega,Hora de entrega

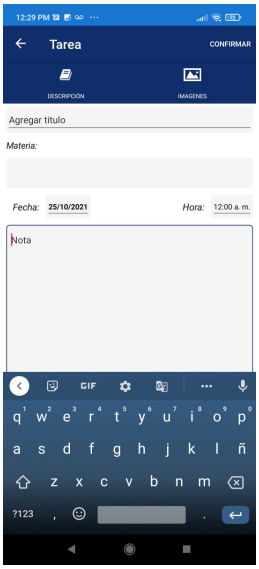


Figura 31: En la opción tarea tenemos cuatro campos para editar: Título, Nota, Fecha de entrega, Hora de entrega, Imágenes

Figura 32: Tenemos la opción de agregar un enlace en la clase seleccionada como por ejemplo una reunión en alguna plataforma.

Figura 33: Vista del enlace una vez creado, cualquier ususario puede ingresar a él presionando sobre el mismo.

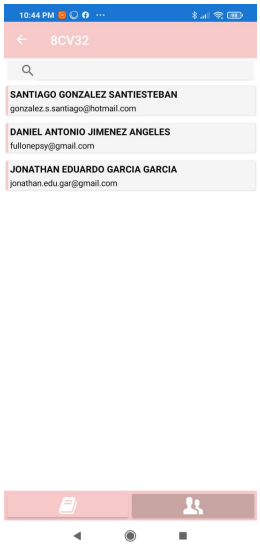


Figura 34: Al seleccionar una materia podemos ver a los alumnos pertenecientes al grupo.

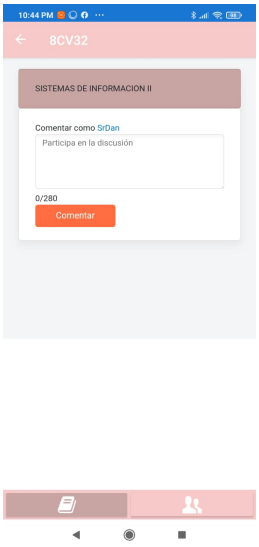


Figura 35: La sección de comunidad permite a los alumnos pertenecientes a alguna de las clases publicar reseñas y comentarios que podrán visualizar otros alumnos pertenecientes al grupo.

10. Conclusión

El presente proyecto de inicio a fin implicó retos importantes de ellos podemos destacar el aprendizaje para aprovisionar dinamicamente los servicios en AWS, la recopilación de los metadatos de cada ejecución, el desarrollo de la aplicación web para presentar la información. Esta aplicación resuelve las necesidades que fueron identificadas durante la etapa de investigación del desarrollo. Durante la etapa de pruebas y puesta a disposición de los pilotos se observó una aceptación positiva por parte de los usuarios y su retroalimentación confirma que esta aplicación es de utilidad en la realización de pruebas de carga. Es importante destacar el apoyo que se tuvo por parte de las materias cursadas en la Maestría en Ingeniería en Seguridad y Tecnologías de la Información y el apoyo de los asesores que dieron las bases en el diseño y construcción de la plataforma.

Referencias

[1] I. SendGrid Labs LLC (SendGrid, *Loader.io*. Estados Unidos, 2012. [En línea]. Disponible: https://loader.io/

[2] G. Labs, *K6*. Suecia, 2017. [En línea]. Disponible: https://k6.io/

[3] I. Dotcom-Monitor, *LoadView*. Estados Unidos, 2000. [En línea]. Disponible: https://www.loadview-testing.com/es/

[4] BrowserStack, *BrowserStack*. India, 2011. [En línea]. Disponible: https://www.browserstack.com/

[5] M. Ortiz., *Ingeniería de Software Modelos de Desarrollo de Software. 2021, de Ingeniería de Software*. India, 2012. [En línea]. Disponible: http://isw-udistrital.blogspot.com/2012/09/ingenieria-de-software-i.html

11. Anexos

- Diseño Pantalla de tareas, 16 jun 2021