Tecnológico Nacional de México
Instituto Tecnológico de Tijuana

Subdirección Académica
Departamento de Sistemas y Computación
Ingeniería en Sistemas Computacionales
Semestre: Agosto - Diciembre 2021

DISEÑO DE INTERFAZ DE USUARIO UI/UX

ADD - 1703

Proyecto Final: ADDO IMSS

Integrantes:

Moran Meraz Abraham	17212161
Santos Poblete Guillermo Leonardo	17211564
Martinez Trujillo Jesus Israel	162120238
Ruiz Alfaro Eduardo	17211559
Franco Corona Rafael	17211521

Miguel Angel Lopez Ramirez

Campus Tomas Aquino

"Por una juventud integrada al desarrollo de México"

Tijuana B.C. a 23 de noviembre del 2021

Índice

Planteamiento del Problema.

Actualmente el departamento de trasplante de órganos del IMSS de Tijuana no tiene una forma de procesar de manera sencilla la documentación del proceso de trasplante de órganos para su fácil llenado, recopilación y compartición de dichos documentos. Solucionar este problema tendría como consecuencia una mejora en el tiempo necesario para llevar a cabo este proceso de llenado y obtención de los documentos dando así una mejora de la calidad de vida en el trabajo del doctor y las personas relacionadas a este departamento.

Introducción.

El proyecto a realizar es una aplicación dirigida hacia un doctor del IMSS para apoyar parte de su trabajo del día a día, en la realización de algunos documentos o formatos para sus pacientes que quieren donar alguno de sus órganos. Para esto el doctor podrá ingresar a su cuenta, y dentro de esta él podrá realizar las acciones de añadir algún paciente nuevo o poder trabajar en los documentos de un paciente ya existente.

Dentro de algún paciente, se podrá realizar la creación de los documentos, rellenar formatos pre-construidos, rellenar múltiples documentos con alguna característica diferente, y podrá verificar cuales son los documentos que ya están rellenados, cuales no lo están, y cuales opciones tienen múltiples documentos ya rellenados con alguna característica diferente.

Otra de las funcionalidades que tendrá esta aplicación será la de compartir estos documentos ya rellenados hacia alguna aplicación que pueda manejar el envío de documentos (Whatsapp, Google Drive, Gmail, etc.) y estos podrán ser compartidos con personas que el piense, sea necesario enviar este tipo de formatos ya rellenados.

Una última funcionalidad que podrá ayudar al doctor, será la forma de tomar fotografías a documentos en físico ya rellenados y estos se transformen en documentos escaneados para poder hacia enviarlos de manera limpia y entendible, con esto se podrá guardar en algunas de las opciones de formatos pre-llenados.

Teniendo en cuenta estos datos, realizaremos las plantillas o vistas de la interfaz de esta aplicación, tomando como apoyo la información obtenida durante la clases de la materia “Diseño De Interfaz De Usuario UI/UX”, en la aplicación gratuita de Figma, para poder tener la idea principal de la interfaz y después implementarla en el desarrollo del proyecto.

En esta documentación, podremos ver el objetivo general de este proyecto, al igual que sus objetivos específicos para lograr el general, daremos un vistazo al desarrollo de este proyecto, explicando cómo fue que pudimos dar a acabo cada parte del trabajo al momento de crear la interfaz, que fue lo que tomamos en cuenta y la retroalimentación dada por el doctor sobre lo que él podía entender y que sería mejor para él, para el uso diario de la aplicación, y cuáles fueron las pruebas hechas y los resultados finales de todo este desarrollo.

Objetivos

Objetivo General.

Desarrollo de un interfaz gráfica funcional para la administración y gestión de expedientes para pacientes donantes de órganos, esta se basará en formularios capaces de hacer las funciones de un gestor de archivos, como lo es el editar, agregar y eliminar archivos, esto con el fin de agilizar el llenado de estos documentos, el sistema contará con la funcionalidad de compartir dicha documentación mediante un dispositivo móvil para su manejo práctico.

Objetivos Específicos.

- Diseño de prototipos de navegación de interfaz de la aplicación
- Diseño de una interfaz intuitiva para el usuario
- Diseñar una interfaz y experiencia de usuario apta para tareas repetitivas
- Diseñar una interfaz funcional que pueda ser utilizada por el usuario
- Diseño y desarrollo de aplicación para plataformas de IOS y Android
- Implementar el almacenamiento de documentos en base de datos mediante el escaneo de los formatos
- Desarrollo e implementación de módulos de autollenado para aquellos formatos que lo requieran
- Diseño de módulos para la inserción y modificación de datos del paciente.

Desarrollo del Proyecto.

Diseño de la aplicación móvil

Para el desarrollo de los mockups de la interfaz gráfica de la aplicación móvil se utilizó la aplicación web Figma las cual nos permitió hacer modelados preliminares que se presentaron al interesado para su retroalimentación y verificación de forma iterativa.

Figura x etapa temprana del diseño, desarrollado en Figma

En cada una de estas iteraciones se presenta el diseño, el interesado lo revisa y proporciona su retroalimentación y se trabaja en correcciones a partir de esta.

Figura x mejoras en el diseño basado en retroalimentación de usuarios

Las funcionalidades a desarrollar como lo son el escaneo de documentos también pueden ser mostradas a través de esta herramienta de diseño para proporcionar de una forma clara una previsualización del resultado al que se desea llegar con el proyecto.

Figura x escaneo de documentos por medio de la cámara.

Módulo para ingreso y actualización de datos del paciente

Para el desarrollo del módulo de “Paciente” que nos permite editar, e ingresar pacientes se usó la Figura X como screen de la aplicación, está contiene multiples inputs que nos permiten llenar y posteriormente registrar los datos en la base de datos por medio del

servidor conecto a la app, esté screen se usa para ambos propósitos, el crear un nuevo paciente y el editar los datos que se tienen registrados. Para realizar la creación de un nuevo paciente usamos un método post de la API que implementa el servidor, el cual es usado al momento de realizar el Submit con los valores requeridos por parte del frontend.

```
//Código del screen .ts, método usado al presionar el botón donde
válida si tiene los valores requerido y posteriormente llamar al método
post de la api.
onSubmit() {
    if (!this.patienteForm.valid) {
      return false;
    } else {
      this.api.createPaciente(this.patienteForm.value).subscribe((res)
=> {
        this.zone.run(() => {
          this.patienteForm.reset();
          this.router.navigate(['/tabs/buscar-paciente']);
        });
      });
    }
  }
//Código del método post que está en un servicio del frontend que
contiene todos métodos de la api
createPaciente(paciente: ApiPaciente): Observable<any> {
    return this.http.post<ApiPaciente>(
      `${this.url}/paciente`,
      {paciente, formatos},
      this.httpOptions
    );
```

Figura x codigo del modulo de paciente

Por parte del servidor se tiene un modelo que contiene la estructura de los datos que tendrá cada variable y el cómo estará formado el documento a guardar en la base de datos.

```
export class MyForm {
  @prop()
  public id_format?: string;
  @prop()
  public name?: string;
  @prop()
  public isUpload?: boolean;
}
export class ApiPaciente {
  readonly _id: Schema.Types.ObjectId;
  @prop({ required: true })
  name: string;
  @prop({ required: true })
  apellido_p: string;
  @prop({ required: true })
  apellido_m: string;
  @prop({ required: true })
  nss: string;
  @prop({ default: 'Muerte encefálica' })
  diagnostica: string;

  @prop({ type: () => MyForm })
  public formatos: MyForm[];
}
```

Figura x modelado de la estructura de datos del paciente

Para realizar la lógica que permitirá insertar un nuevo documento se implementa la siguiente sección de código que contiene el método async que permite guardar los datos en la base de datos mongodb.

```
async create(createPacienteDto: {
  name: string;
  apellido_p: string;
  apellido_m: string;
  nss: string;
}) {
```

Figura x modelo de creación de pacientes

Una vez codificado el modelo y el método a utilizar para guardar un nuevo documento, se implementa el código el controller que permite realizar el correspondiente post en una ruta determinada, en el caso de la inserción la ruta es la raíz.

```
@Post('')
async addPaciente(@Res() res, @Body() body) {
  const generated = await this.pacienteService.create(body);
  return res.send(generated);
}
```

Figura X metodo post asincrónico

Haciendo uso de la interfaz del modelo de los datos que se usó en la inserción del paciente ahora se utiliza para editar los datos de un paciente previamente registrado. Con la interfaz de editar paciente se puede observar que los datos se rellenan para tener la información que se tiene en la base de datos. Esto a través de llamar al método “get Paciente” que nos proporciona los datos al inicial el componente que hace referencia a la pantalla edita. En está pantalla se cambia el contenido si se requiere del paciente y con el botón se realiza el update de los datos en el servidor. El código a continuación muestra el método al realizar click.

```
async updateForm() {
    if (!this.updatePacientForm.valid) {
      return false;
    } else {
//Método para llamar el metodo post para mandar al servidor la peticion
      await this.api
        .updatePacient(this.paciente, this.updatePacientForm.value)
        .subscribe((res) => {
          console.log(res);
          this.updatePacientForm.reset();
```

Figura x metodo para actualizar forma de paciente

Por el lado del servidor el controlador contiene un método http post para realizar la actualización de la información, el cual retorno un httpstatus con un ok si se realizó de manera correcta.

```
@Post('update-paciente/new/:paciente')
   async updatePaciente(@Res() res, @Body() body, @Param('paciente')
paciente) {
    await this.pacienteService.updatePaciente(paciente, body);
    return res.status(HttpStatus.OK).json({ msg: 'Actualizado paciente'
});
  }
```

Figura x metodo post para enviar datos de paciente

El código responsable de editar la base de datos es el “updatePaciente” de controlar, el siguiente código muestra el uso de métodos de mongoose para editar los datos.

```
async updatePaciente(paciente, data) {
    await this.pacienteModel.findByIdAndUpdate(paciente, {
```

Figura x metodo updatePaciente

En este caso se usaron métodos que se implementaron previamente en otros módulos donde haciendo uso del método “getPaciente” que se usa al inicializar el screen se puede obtener los datos del paciente de forma sencilla. En este caso la pantalla para ver los datos simplemente muestra los datos del paciente que existen actualmente en la base de datos. Este simplemente permite visualizar sin ningún tipo de modificación por parte del usuario en está pantalla, para realizar cambio requiere seleccionar otra opción en la pantalla de detalles del paciente. A continuación se mostrará el código respectivo para que funcione mostrando la lógica del componente, del servicio y el controlador del servidor en ejecución.

Método del componente al inicial la pantalla:

```
getPacientData(paciente) {
    this.api.getPaciente(paciente).subscribe((res) => {
      this.datapaciente = res;
    });
  }
```

Figura x metodo getPacientData

Método del servicio que permite obtener los datos del servidor:

```
getPaciente(_id: string): Observable<ApiPaciente> {
    return this.http
      .get<ApiPaciente>(`${this.url}/paciente/${_id}`)
                              .pipe(tap((response:     ApiPaciente)     =>
this.myDataPaciente.next(response)));
  }
```

Figura x metodo get Paciente

Método http get por parte del controlador del servidor:

```
@Get(':id')
  async getPaciente(@Res() res, @Body() body, @Param('id') id) {
    const paciente = await this.pacienteService.getByIdpacientes(id);
    return res.send(paciente);
  }
```

Figura x metodo get Paciente

Método para buscar datos del paciente específico en la base de datos:

```
async getByIdpacientes(id) {
    return await this.pacienteModel.findById(id).exec();
  }
```

Figura x metodo getByIdpacientes

La funcionalidad para eliminar el paciente también involucra la eliminación de todos sus formatos guardados. Para esto no se usa una nueva pantalla, para realizar la eliminación de los datos el usuario debe entrar a los detalles del paciente, y seleccionar el botón de la parte superior derecha y darle click en la opción de “Eliminar datos”. Este botón realiza la eliminación al llamar un método del servicio que permite realizar la petición http delete al servidor. En el código a continuación se muestra los métodos utilizados:

Primeramente se tiene el código para usar el componente de ionic que muestra un modal preguntado la confirmación de la eliminación:

```
async showOptionToDelete() {
    const alert = await this.alert.create({
      header: 'Alerta',
      message:
            '¿Seguro de que quiere borrar al paciente y todos sus
documentos?',
      buttons: [
        {
          text: 'Cancelar',
          role: 'cancel',
          handler: () => {
            console.log('Declined the offer');
          },
        },
        {
          text: 'Aceptar',
```

Figura x componente gráfico para mostrar eliminar

Método del servicio que llama la metodo http delete del servidor:

```
deletePacienteAndFilesRef(paciente: string) {
    return this.http.delete(`${this.url}/paciente/${paciente}`).pipe(
      tap(() => {
        this._refresh$.next();
      })
    );
  }
```

Figura x método para borrar paciente y archivos de paciente

Método para realizar la modificación en la base de datos:

```
async deletePaciente(paciente: string) {
    await this.formatoModel.deleteMany({
      paciente: paciente,
    });
    return await this.pacienteModel.findByIdAndDelete({
      _id: paciente,
    });
  }
```

Figura x metodo borrar paciente de base de datos

Pruebas y Resultados.

Existe la posibilidad de agregar un tipo de estudio en la pantalla de estudios posibles, por ejemplo en la **figura x** podemos observar los pasos de como agregar un nuevo formato con el nombre covid y que al final nos lo muestre como un estudio en el cual puedes subir un documento al servidor.

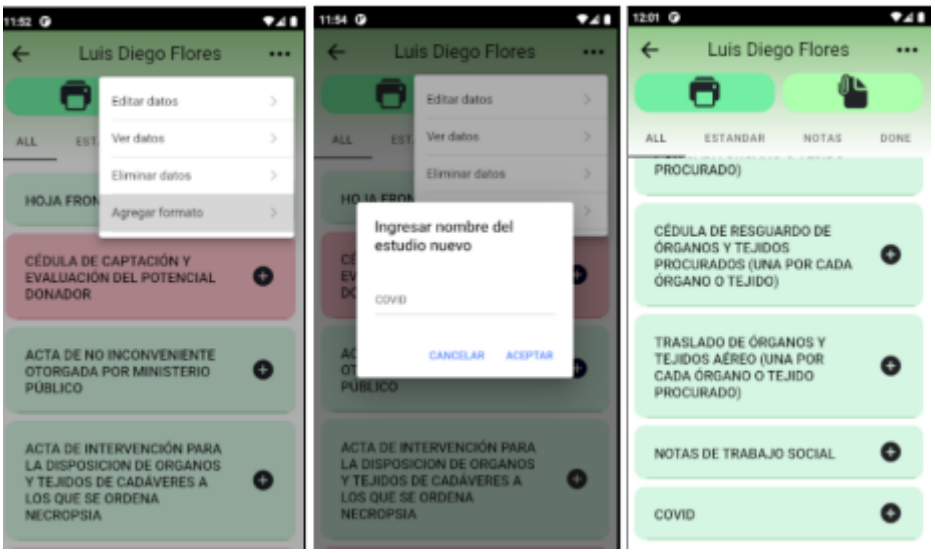
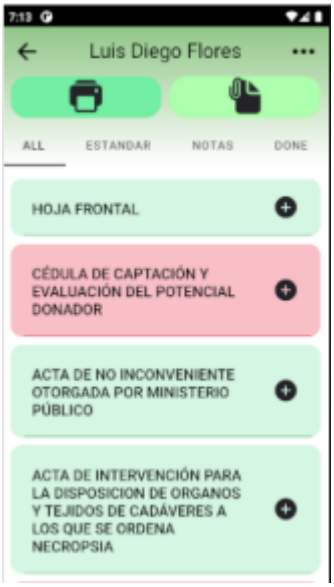


Figura x: Pasos para agregar un estudio.



Para la prueba del escaneo de documentos y su almacenamiento es necesario primeramente seleccionar cual es el tipo del formato que requiere el paciente dependiendo del tipo de estudio que se le va a realizar o se le realizó, esto para que facilite el ingreso de la documentación por parte del doctor que la esté utilizando.

Una vez seleccionado el tipo de estudio es posible ingresar el número de imágenes que se van a escanear para así ahora escanear el documento por medio de una foto que es tomada en el momento y así subirla a la aplicación, teniendo la posibilidad de recortar dicha foto para seleccionar solo la parte deseada y una vez conforme con el resultado esta se subirá a la aplicación, tal como se muestra en la **figura (x)**.

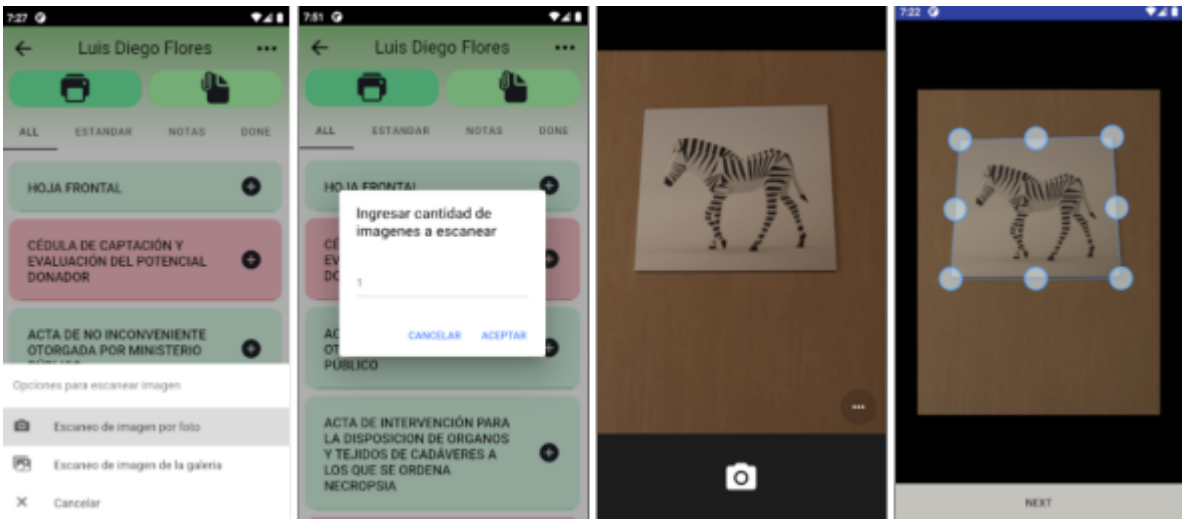


Figura x: Secuencia de pasos de la opción “Escaneo de imagen por foto”.

En caso de utilizar el escaneo de imagen de la galería, será posible seleccionar una imagen que esté presente en la galería de su dispositivo móvil y así mismo recortarla como en el caso anterior, así como se muestra en la **figura (x)**.

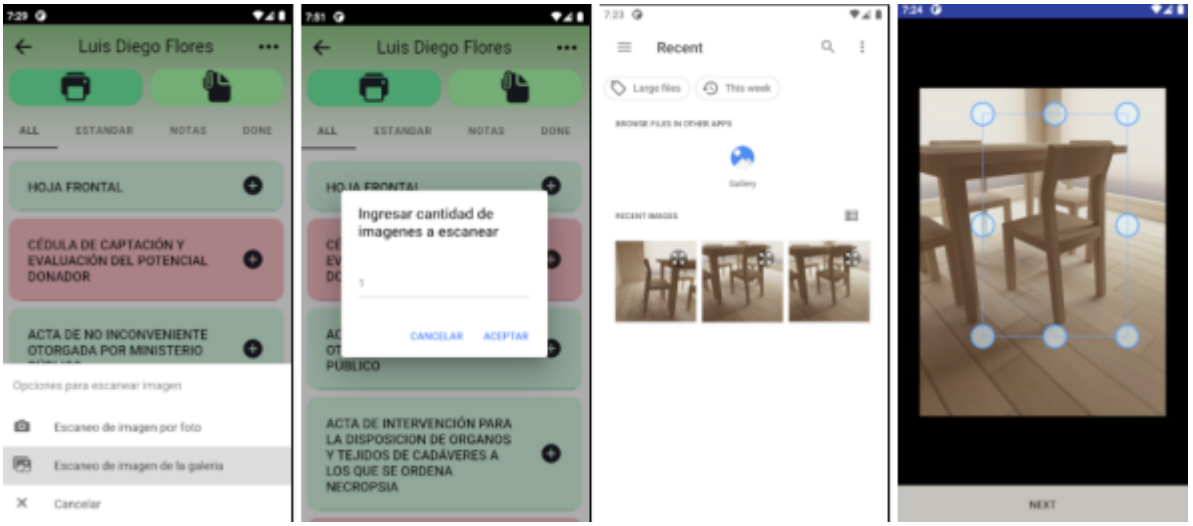


Figura x: Secuencia de pasos de la opción “Escaneo de imagen de la galería”.

Una vez realizado el proceso de las **figuras x y x**, la casilla del estudio seleccionado se mantendrá en rojo, mostrando así que existe un archivo en el servidor de la aplicación. Si se vuelve a seleccionar la misma casilla, podrá observar que ahora las opciones que nos muestran son “Vista del documento” y “Eliminar”, que con ellas podremos ver el documento que fue subido al servidor o eliminarlo respectivamente, así como se muestra en la **figura x**.

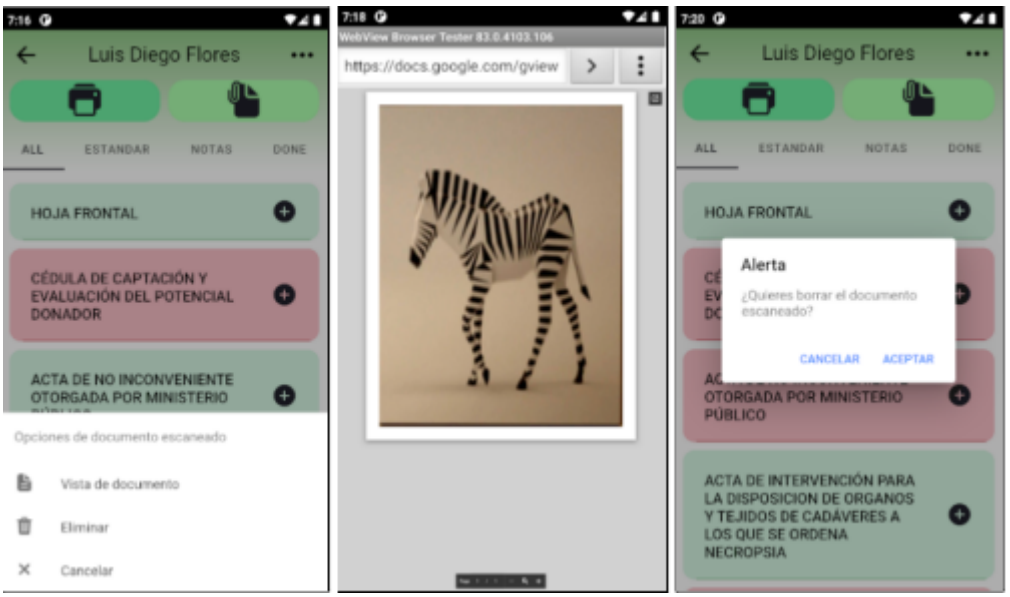


Figura x: Opciones posibles una vez subido el formato.

En la **figura x** podemos observar que al seleccionar los detalles del paciente podemos seleccionar los formatos que existen para el paciente en cuestión para así tener la posibilidad de compartirlo. Seleccionar la opción de compartir abrirá una ventana que nos permitirá seleccionar la opción con la que queremos compartir el documento (como gmail, whatsapp, etc.).

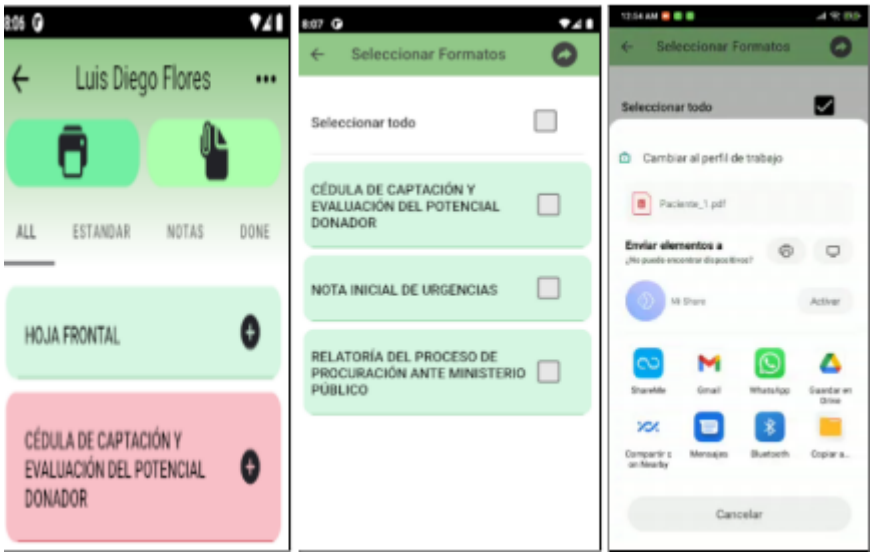


Figura x: Compartición del formato.

Conclusiones.

Al haber finalizado todo el proceso de desarrollo de esta aplicación, podemos concluir, lo que es la realización completa de una interfaz de alguna aplicación, los pasos que debemos de tomar en cuenta y las complicaciones que podemos toparnos durante el progreso que llevemos en este proyecto.

Debemos tomar en cuenta cada parte mencionada por el doctor, en lo que pedía como requerimientos y estos pasarlos a una vista gráfica que sería la interfaz. Podemos darnos cuenta que puede ser un poco complicado llegar con una idea agradable para todos y más para una persona que no conocemos sus gustos y la perspectiva que tiene el de la interfaz que utilizará en su trabajo para que lo pueda ayudar. Son los obstáculos que debemos pasar como desarrolladores al momento de crear alguna aplicación, y que poco a poco debemos encontrar la manera de cómo resolver los problemas que nos encontremos.

Al final, con los aprendizajes que recibimos de la materia, con la información impartida por el profesor, información de documentaciones y parte retroalimentación de algunos estudiantes del ITT al igual que del doctor, nos ayudó a poder terminar en hora buena este proyecto con todas las características que se nos pidieron y terminamos con una aplicación totalmente funcional y con una interfaz bastante profesional para que alguien pueda utilizarla sin problema y pueda trabajar en ella.

Una interfaz no necesariamente se termina hasta que la aplicación es entregada hacia el cliente, es lo contrario. Una aplicación puede tener una evolución de la interfaz, dada de la mano con las nuevas tecnologías para el UI/UX que salen día con día. Con el tiempo estas nuevas tecnologías ayudan mucho a desarrolladores de front-end para llevar a cabo la realización de interfaces cómodas, sencillas, legibles y agradables estéticamente hacia los usuarios para que no haya problemas al momento de utilizarlas. Con esto podemos saber, que este proyecto con el tiempo se podrá modificar la interfaz principal para que pueda utilizar estas tecnologías y poderla llevar todavía más lejos.

Será cuestión del tiempo en el desarrollo de nuevas tecnologías para los desarrolladores de front-end para que esto pueda ser verdad y podamos utilizarlas a nuestro beneficio.

Bibliografía