Taller de Proyecto II
2020

PC. 6 - Combi IoT

Informe Final

Grupo de Desarrollo

- Giammarini, Paula Andrea 1218/8
- Trybus, Magdalena 1652/3
- Zappettini, Victoria 1607/7

7 de Diciembre 2020

Índice

1. Proyecto

El proyecto Combi-IoT tiene como objetivo mejorar la calidad de la experiencia de los usuarios con el servicio de la combi, a través de un sistema que permita:

- **Reservar pasajes** en una determinada fecha y hora elegida por el usuario, el cual accederá a la misma mediante un dispositivo con acceso a Internet. Una vez realizada la reserva, la aplicación le retornará un **código pdf417** el cual debe presentarse en la combi antes de viajar para validar su boleto.

- **Leer el código** que tiene del usuario. Este código será **descifrado** para obtener la información correspondiente a la reserva y permitirá verificar si la misma es válida y, así, acreditar o no el viaje del usuario.

Una decisión adicional que se tomó con el equipo fue no utilizar un almacenamiento en la nube para la base de datos.

Inicialmente el proyecto estaba pensado para que la decodificación del código PDF417 se realice desde una cámara USB conectada a una raspberry pi desde la combi. Sin embargo por la **pandemia** de público conocimiento que nos tocó atravesar, realizamos la **decodificación vía web**.

2. Materiales y Presupuesto

A continuación se muestra una tabla con los materiales necesarios para el desarrollo del proyecto en un **contexto normal** junto con sus precios aproximados:

Materiales	Costo aproximado [ARS]
Raspberry pi 3 Model B	$ 6500
Cámara web USB	$ 3500
Powerbank (opcional)	$ 2000
Módulo GPS (opcional)	$ 4000
Panel LED	$ 300

Los valores se obtuvieron a través de la página web de Mercado Libre Argentina, al día 9/10/2020. Los valores pueden ser cambiantes debido a la devaluación de la moneda Argentina (https://www.mercadolibre.com.ar).

Debido al contexto de pandemia y a que la decodificación la realizamos vía web, los materiales mencionados anteriormente no son necesarios. Solo se necesita contar con una computadora de escritorio o notebook con acceso a internet.

3. Descripción del Proyecto

Para la implementación del proyecto se desarrolló **una página web de reservas de turnos** para una combi. Esta página se programó en Python y html, utilizando el framework web django, el cual sigue el paradigma MTV (model/template/view).

Las dos partes fundamentales del proyecto son: la ***codificación*** y la ***decodificación*** *de un código PDF417*.

Codificación:

El usuario, mediante un dispositivo con conexión a internet, ingresará en el sistema y reservará su pasaje para una fecha y hora determinada. Si la reserva resulta exitosa, se le devuelve al usuario un código PDF417 que representará su boleto. Los datos de la reserva son almacenados en el sistema mediante una base de datos relacional (sqlite3).

La página codifica la imagen a partir de la información ingresada en el formulario de reserva mediante la siguiente función:

```
def codificador(data, reserva):
    # Convierte palabras a código
    codigo = encode(str(data), security_level=7)
    # Genera un código de barras como imagen
    imagen = render_image(codigo)
    imagen.save(MEDIA_ROOT + '/barcode.png')
    reserva.codigo = MEDIA_ROOT + '/barcode.png'
    reserva.save()
```

La función que se muestra arriba permite la codificación de información recibida como parámetro (“data”), haciendo uso de la librería ***pdf417gen***.

Esto lo realiza mediante 3 pasos:

1. Codificación de información: mediante la función **encode** propia de la librería, la cual recibe la información como string y, opcionalmente, dos parámetros que indican la cantidad de columnas y el nivel de seguridad (columns y security_level respectivamente).

2. Generación de la imagen: mediante la función **render_image** se genera la imagen que representa el código previamente obtenido en el punto 1.

3. Guardado de la imagen: mediante la función **save**, la cual recibe una dirección para guardar la imagen generada en el punto 2, y posteriormente se asigna la imagen al campo correspondiente al objeto “reserva” previamente creado.

Para más información consultar la página explicativa de la librería pdf417gen en nuestra wiki: Librería pdf417 + prueba

Decodificación:

Una vez generada la reservación al hacer clic en el botón "**Decodificar imagen**" se realizará la obtención de información, a partir de la imagen del código PDF417, la cual contiene todos los datos de la reserva. En este punto, el usuario tendrá a su disposición la información que ingresó para su reservación.

La página realiza esta funcionalidad mediante el siguiente código:

```
def decodificar_2():
    reader = BarCodeReader()
    results = reader.decode(MEDIA_ROOT + '/barcode.png')
    datos = results[0]['parsed'].replace("'", "\"")
    return ast.literal_eval(datos)
```

Esta función permite la decodificación de una imagen de un código PDF417, haciendo uso de la librería ***pyzxing***. Esta librería es una extensión para Python de la librería **zxing** en Java.

La decodificación se realiza mediante 4 pasos:

1. Crea una instancia del objeto **BarCodeReader,** el cual descifrará los datos.
2. Luego se ejecuta la función **decode**, la cual recibe una dirección a la imagen a decodificar. Generando como resultado una lista que contiene en su primer posición un diccionario con los siguientes campos:
    - filename
    - file
    - format
    - type
    - raw
    - parsed
    - points
3. De los campos nombrados anteriormente, el campo "**parsed**" es el que contiene la información obtenida de la imagen. Por lo tanto en datos nos quedamos con ese campo.
4. Luego se devuelven los datos como un **diccionario** para facilitar la posterior visualización de la información.

Identificación de las partes del proyecto:

a. **Alimentación del dispositivo/placa de desarrollo:**
   En un contexto normal, utilizando hardware físico, la Raspberry pi 3 model b se alimentaría por un powerbank conectado a su entrada de alimentación. Sin embargo, como se explicó anteriormente, en este proyecto no se utilizó.

b. **Conexiones de E/S de la placa de desarrollo con el exterior excepto PC**
   Al igual que en el inciso a, en un contexto normal, sería una conexión de una cámara USB a la raspberry por uno de sus puertos disponibles para la lectura del código pdf417. Sumando a una matriz led que indicaría al usuario la validez de su pasaje. Sin embargo, como se explicó anteriormente, en este proyecto no se utilizó.

c. **Comunicaciones de la placa de desarrollo con la PC**
   No se necesita tener ninguna comunicación con la PC u otro dispositivo adicional a los mencionados en los puntos anteriores.

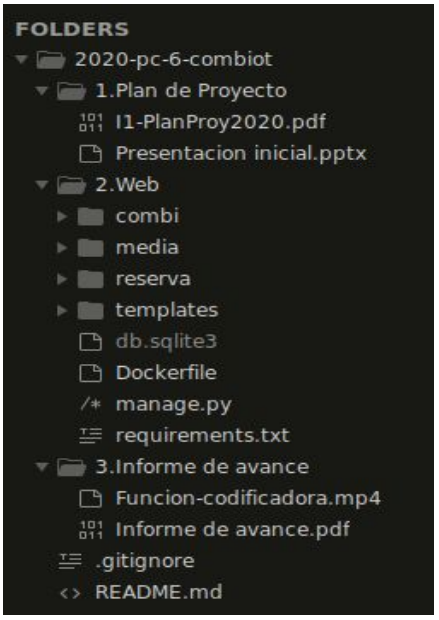
d. **Sistema/interfaz web**
   Interfaz en la que el usuario se encuentra habilitado para reservar un turno, es decir, sacar un pasaje para una combi en una fecha y hora determinada. Este pasaje se representa con un código pdf417 generado por el sistema.

e. **Infraestructura de software en PC**
   Desarrollaremos una página web que se conecta con una base de datos relacional, sqlite.

Explicación del código desarrollado

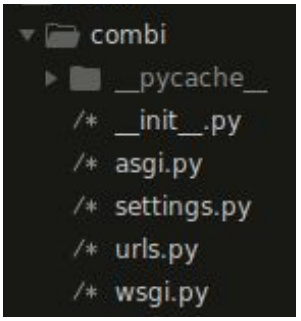
La página web se desarrolló utilizando el lenguaje Python y html, e implementando el framework web django.

En la imagen de la derecha se puede observar la estructura del proyecto que se encontrará en el sistema de control de versiones Github.



La carpeta **2.Web** cuenta con el código fuente de la página web. Dentro de la misma se encuentran diferentes carpetas y archivos que se detallarán a continuación:

- **combi**



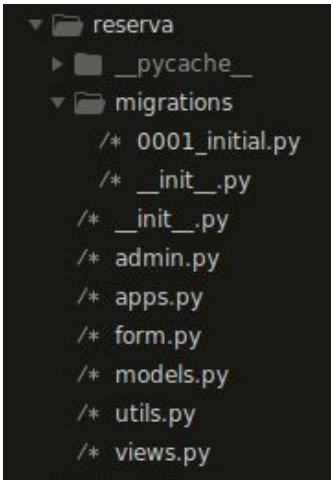
Esta carpeta contiene todas las configuraciones necesarias para que el proyecto Django funcione correctamente. Además en el archivo urls.py se pueden encontrar las distintas direcciones habilitadas de la página web (‘/’, ‘/reserva’, ‘/informacion’, ‘/nueva_reserva’, ‘/decodificacion’, ‘/admin’)

- **media**

Esta carpeta es la encargada de almacenar la imágen del código pdf417.

- **reserva**

Este módulo se compone de varias partes que serán explicadas a continuación:



➢ **models.py**

Este archivo contiene el modelo Reserva que representa a la tabla Reserva en la base de datos. En la siguiente imagen pueden observarse todos los campos que representan a una reserva en el sistema:

```python
from django.db import models


class Reserva(models.Model):
    codigo = models.ImageField(upload_to='')
    nombre = models.CharField(max_length=20)
    apellido = models.CharField(max_length=20)
    DNI = models.IntegerField()
    fecha = models.DateField()
    email = models.EmailField(max_length=254)
    hora = models.TimeField()

    def __str__(self):
        return str(self.apellido) + '-' + str(self.email) + ' ' + str(self.fecha)

```

➢ **migrations**

Esta carpeta contiene las migraciones, es decir, archivos donde se mantiene el registro de los cambios que se realizan en los modelos para mantener sincronizada la base de datos.

➢ **apps.py**

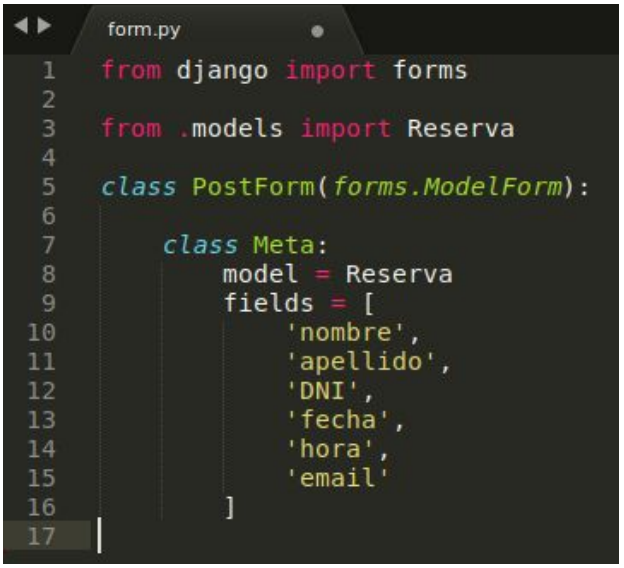
Este es un archivo de configuración de la app **reserva**. Para poder agregarla posteriormente al proyecto Django.

➢ **admin.py**

Este es un archivo de configuración para registrar el modelo **reserva** y poder visualizarlo desde el administrador que proporciona Django.

➢ **form.py**

Este archivo es un formulario de Django. Permite el ingreso de información en el formulario de reserva de la página web y contiene los campos propios del modelo Reserva excepto por el **código** el cual es generado automáticamente por la página.



➢ **utils.py**

En este archivo se encuentran las funciones de **codificación** y **decodificación** explicadas anteriormente. Además contiene una función adicional de decodificación utilizando un decodificador externo.

Para más información dirigirse a la página de la wiki:
Prueba de decodificación

➢ **views.py**

En este archivo están las funcionalidades de la página web.

views.py

```python
from django.shortcuts import render
from reserva.utils import decodificar_2, codificador
from .form import PostForm
from .models import Reserva
from django.utils import timezone
from datetime import datetime


def home(request):
    return render(request, 'home.html', context={})


def decodificar(request):
    # data = decodificar_1() #Alternativa para decodificador online externo
    data = decodificar_2()
    return render(request, 'decodificacion.html', context={'data': data})


def reservar(request):
    return render(request, 'nueva_reserva.html', context={})


def informar(request):
    return render(request, 'info.html', context={})

```

```python
def post(request):
    if request.method == "POST":
        form = PostForm(request.POST)
        if form.is_valid():
            nueva_reserva = form.save(commit=False)
            exists = Reserva.objects.filter(
                apellido=nueva_reserva.apellido,
                nombre=nueva_reserva.nombre,
                DNI=nueva_reserva.DNI,
                email=nueva_reserva.email,
                fecha=nueva_reserva.fecha).exists()
            if exists:
                return render(request, 'repetida.html', context={})
            elif datetime.strftime(nueva_reserva.fecha, '%Y-%m-%d') < timezone.localtime(timezone.now()).date().isofo
                return render(request, 'error_fecha.html', context={})
            else:
                nueva_reserva.save()
                reserva = Reserva.objects.get(
                    fecha=nueva_reserva.fecha,
                    email=nueva_reserva.email,
                    nombre=nueva_reserva.nombre,
                    apellido=nueva_reserva.apellido,
                    DNI=nueva_reserva.DNI,
                    hora=nueva_reserva.hora)
                data = {
                    'nombre': reserva.nombre,
                    'apellido': reserva.apellido,
                    'DNI': reserva.DNI,
                    'fecha': str(reserva.fecha),
                    'hora': str(reserva.hora),
                    'email': reserva.email,
                }
                codificador(data, reserva)
                data['imagen'] = reserva.codigo
                reserva.save()
                return render(request, 'reserva_exitosa.html', context={'data': data})
    return render(request, "home.html", context={})

```

En la **primera imagen** se encuentran 3 funciones que se encargan simplemente de redirigir la request del usuario al archivo html que corresponde para, de esta forma, mostrar la información correcta en cada caso. Estas funciones son: **home** (redirige a la página principal), **reservar** (redirige al formulario de reserva) e **informar** (redirige a la página de información).

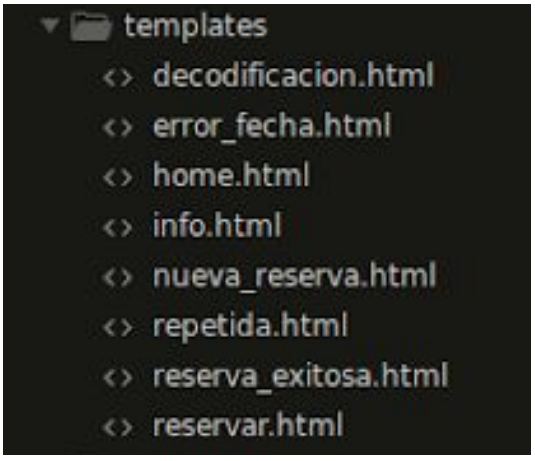
La función **decodificador s**e encarga de invocar a la función **decodificar** explicada anteriormente, para obtener los datos de la imagen del pdf417 y, posteriormente, envía la información obtenida a la página donde se muestra la decodificación exitosa.

En la **segunda imagen** se visualiza la funcionalidad **post**. Esta función recibe la información ingresada en el formulario de reserva y verifica que la misma no esté repetida y sea válida (con fecha igual o mayor a hoy). En caso contrario redirige a la página de error correspondiente.

Si la información es correcta, se guardan los datos de la reserva y se envían a la función **codificador** para que sean cifrados en la imagen pdf417. Luego se redirige a la página que muestra que la reserva fue exitosa.

- **templates**

Esta carpeta contiene los archivos html que representan las vistas de la página web.



- **db.sqlite3**

Este archivo contiene la base de datos local generada al correr las migraciones.

- **Dockerfile**

  Imagen docker que podría ser utilizada si se simulara una raspberry pi o en el hardware real. Para más información al respecto dirigirse a la página de la wiki: Prueba con docker

- **manage.py**

  Este archivo se utiliza para ejecutar comandos de administración del proyecto, y permite correr el servidor.

- **requeriments.txt**

  Este archivo contiene todas las librerías y dependencias necesarias para el correcto funcionamiento del proyecto.

Para mayor detalle, dirigirse a los siguientes links:

➔ Bitácora
➔ Código fuente - repositorio github

4. Guía de Instalación: Proyecto y Ambiente de Desarrollo

a. Ambiente de desarrollo

El proyecto está desarrollado para ser ejecutado en el sistema operativo linux. A continuación se detalla el software necesario para preparar el ambiente de desarrollo junto con los comandos para su instalación. Dichos comandos deben ser ejecutados en una terminal Linux:

- **python 3 y pip:**

```
sudo apt-get install python3-pip
```

- **git:**

```
sudo apt-get install git
```

- **virtualenv:**

```
sudo apt-get install python-virtualenv
```

- **un navegador:** puede ser google Chrome, mozilla Firefox, o cualquiera con el que cuente la computadora.

Para ver, modificar, mantener y/o mejorar el código fuente del programa solo se necesita de un programa **editor de texto**, ya que el proyecto está desarrollado en python y html. Se sugiere la utilización de Visual Studio Code o Sublime text. El editor de texto puede ser descargado desde el gestor de aplicaciones de linux.

Aclaración: todos los pasos nombrados en esta sección son para la distribución Ubuntu de linux.

b. Proyecto

Guía rápida para instalar el proyecto

1) Descargar el script de instalación desde este link.

2) Abrir una terminal y posicionarse (utilizando el comando **cd**) en la carpeta donde se encuentra el script descargado (instalador.sh)

3) Luego ejecutar el script mediante el comando:

```
sh instalador.sh
```

o mediante este otro comando:

```
bash nombre_de_archivo.sh
```

Este último paso puede demorar unos minutos.

Al finalizar se tendrá instalado todo lo necesario y se encenderá el servidor que corre al proyecto de forma local.

Guía detallada para instalar el proyecto

1) Antes de clonar el repositorio, se recomienda crear un *entorno virtual*. Eso creará un ambiente aislado para el proyecto. Esto significa que el proyecto puede tener sus propias dependencias, independientemente de las dependencias que tenga cualquier otro proyecto en su sistema.

Aquí es donde se instalarán los requisitos que hacen que el proyecto funcione correctamente.

Para ello, ejecutar estos comandos en la ventana de la Terminal en linux:

```
virtualenv entorno                    # crea el entorno virtual
. entorno/bin/activate                # activa el entorno
```

2) Para clonar el repositorio seguir los siguientes comandos:

```
mkdir proyecto
cd proyecto
git init
git clone https://github.com/tpII/2020-pc-6-combiot.git
```

3) Una vez realizado el paso 2 podrás ver la carpeta **2020-pc-6-combiot**, la cual contiene el proyecto, en tu sistema.

Luego, con el entorno activado, instalar los requerimientos mediante los siguientes comandos:

```
cd 2020-pc-6-combiot/2.Web/
pip install -r requirements.txt
```

4) Al finalizar la instalación de los requerimientos, se deben correr las migraciones para obtener la base de datos local ejecutando el comando:

```
python manage.py migrate
```

5) Finalmente, para correr localmente el servidor y hacer uso de la página web utilizar el comando:

```
python manage.py runserver
```

5. Problemas y Soluciones

Una de las limitaciones más significativas para el proyecto fue no contar con el hardware para la realización del mismo. Es por ello que por falta de material y el corto tiempo, la **decodificación del código PDF417** se realizó vía Web.

Por otra parte, en cuanto a la investigación de librerías para generar un código PDF417 y para decodificarlo, se puede mencionar que:

- ➔ Generación del código PDF417: la búsqueda fue sencilla y también su implementación.
- ➔ Decodificación del código PDF417: luego de una búsqueda exhaustiva se encontraron algunas librerías como:
  - pyzbar (Python con C)
  - pyzxing (Python con Java)
  - barcode-read (C#)
  - licencia de Dynamsoft (necesitaba licencia)

Sin embargo, la mayoría, provocaron más inconvenientes que soluciones.

Luego de una reunión con el docente, se plantearon dos alternativas: **decodificador online externo** o utilizando una **imagen docker junto con la librería pyzxing**. Al realizar las pruebas aisladas ambas soluciones funcionaron y todo anduvo correctamente. Se optó por usar la imagen docker.

Sin embargo, investigando más y al integrarlo al sistema con Django, se descubrió que Django permite ejecutar aplicaciones y herramientas escritas en el lenguaje Java. Esto significó que la imagen docker no sería necesaria y fue posible simplemente utilizar la librería pyzxing.

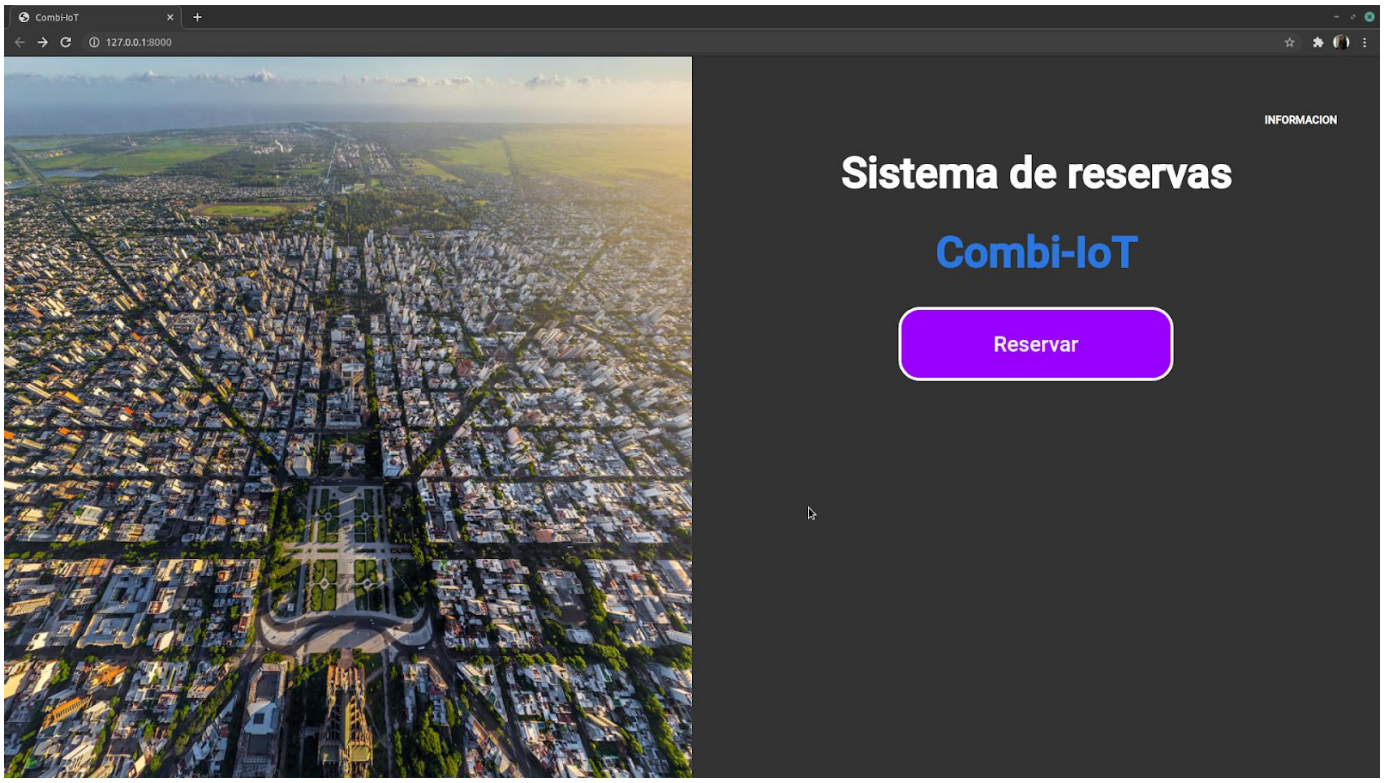
Si bien se utilizó esta última, las alternativas, al igual que la imagen docker, están disponibles en el código fuente del proyecto.

A modo de **trabajo futuro** sería bueno incorporar al proyecto:

- ➔ Implementación en hardware físico o simulado: se necesitará una raspberry pi 3 model b y una cámara usb si se quiere implementar físicamente. En el caso de la simulación podría utilizarse la imagen docker que está presente en el proyecto y adaptarla con una máquina virtual que simula una raspberry.
- ➔ Notificaciones: se requerirá modificar o añadir funcionalidad en el código fuente de la página web.
- ➔ Utilizar un módulo GPS y mostrar recorrido por la web: se requerirá modificar o añadir funcionalidad en el código fuente de la página web. Además de añadir el módulo GPS a la raspberry pi y configurarlo.
- ➔ Decodificar imágenes con cámara web: una librería que podría utilizarse es Opencv de python. Además de añadir la cámara web a la raspberry pi y configurarla.
- ➔ Implementar usuarios en la página web: se requerirá modificar o añadir funcionalidad en el código fuente de la página web.

6. Documentación en Formato Gráfico y Video

Página Principal de la web de reservas:



Si se hace clic en "**INFORMACIÓN**" se redirige a una página donde se cuenta un poco sobre el proyecto Combi-IoT 2020.

Si se presiona el botón "**Reservar**" se redirige a un formulario para la reserva.

Luego de completar los datos, y presionar el botón "**Enviar datos**", si los datos son correctos, se generará una reservación. Junto con la reservación ***se genera el código PDF417*** que la representa, el cual estará disponible para que el usuario lo descargue.

El código PDF417 se puede descargar al presionar el botón de descarga o bien, haciendo clic en la imagen.

Si se presiona el botón "**Decodificar imagen**" se muestran los datos completos de la reserva que son el resultado de la ***decodificación de la imagen del código PDF417***.

Video de prueba del sistema

7. Conclusiones

Para concluir, se considera como aporte del proyecto realizado, en comparación con el proyecto Combi-IoT que se continuó, haber logrado la *codificación y decodificación de un código PDF417*. Sin embargo, por haber transitado durante todo el cuatrimestre el aislamiento social preventivo y obligatorio (ASPO) y no haber concurrido a la facultad, no pudimos implementar el proyecto con el hardware que corresponde lo cual hubiese aportado otro significado al mismo en cuanto a lo que la materia representa.

En cuanto a la experiencia de las autoras de este proyecto, se valora la realización del proyecto como aprendizaje previo a la entrada del mundo laboral. Fue de utilidad para aplicar conceptos teóricos aprendidos durante la carrera y la conexión existente entre ellos. Otros aprendizajes a destacar fueron organizar los tiempos para realizar las entregas apropiadamente en tiempo y forma, y realizar investigaciones acordes a las necesidades que surgían a lo largo del proyecto.