

Benemérita Universidad Autónoma de Puebla

Materia: Servicio Social

Nombre del profesor: Alberto Roman
Flores

Fecha de entrega : 06 de Julio del 2025

Nombre y Matricula del alumno:

Aplicación Móvil para Examen

Tamanis Rodríguez Pavel 202058576

Tabla de contenido

<i>Portada</i>	1
<i>Introducción</i>	3
<i>Descripción general del Servicio Social</i>	3
<i>Código</i>	5
<i>Anexos</i>	7
<i>Conclusión</i>	17
<i>Hoja de Firmas</i>	18

Introducción

La concepción de este proyecto surgió de mi interés por desarrollar una aplicación móvil, impulsado particularmente por la innovación y el ecosistema de los productos Apple. Esto me llevó a adquirir una Mac M1 Air, con el objetivo de sumergirme en su entorno de desarrollo y experimentar con sus modernas tecnologías de programación. A lo largo de este reporte, se detallará el proceso de creación de la aplicación, esperando que las lecciones aprendidas y la metodología descrita puedan resultar valiosas para el lector.

Descripción general del Servicio Social

2.1. Desarrollo de una aplicación móvil para generar un examen

- Nombre formal del proyecto: "Desarrollo de Aplicación Móvil Educativa en Swift sobre Fundamentos de Programación".

2.2. Objetivos del Servicio Social

2.2.1. Objetivo General

Contribuir al fortalecimiento del aprendizaje de los fundamentos de programación en estudiantes o entusiastas de nivel básico, mediante el desarrollo de una aplicación móvil interactiva y accesible.

2.2.2. Objetivos Específicos

- Diseñar y desarrollar una aplicación móvil nativa para iOS utilizando el lenguaje de programación Swift.
- Implementar una base de datos interna con 50 preguntas de opción múltiple (3 opciones por pregunta) sobre temas básicos de programación (variables, tipos de datos, estructuras de control, etc.).
- Crear una interfaz de usuario intuitiva y amigable que facilite la navegación y la interacción del usuario con el cuestionario.
- Proporcionar retroalimentación inmediata al usuario sobre la corrección de sus respuestas.
- Aplicar los conocimientos teóricos adquiridos en la carrera en un proyecto práctico con impacto social o educativo.

2.3. Descripción del Proyecto Desarrollado: Aplicación Móvil Educativa

2.3.1. Justificación y Alcance

- **Justificación:** La necesidad de herramientas móviles para el estudio autodidacta, reforzar conocimientos de forma lúdica.
- **Alcance::** La aplicación se enfoca en preguntas de opción múltiple sobre conceptos teóricos básicos y no incluye un compilador de código ni ejercicios de programación complejos. Está diseñada para iOS.

2.3.2. Público Objetivo

- Estudiantes de primeros semestres de carreras de informática o afines, personas autodidactas interesadas en aprender los fundamentos de la programación, o cualquier individuo que desee repasar conceptos básicos de manera interactiva.

2.3.3. Características Principales de la Aplicación

- Detalla las funcionalidades clave de tu app.
- Cuestionario interactivo con 50 preguntas únicas.
- Cada pregunta presenta 3 posibles respuestas, siendo solo una la correcta.
- Los temas cubiertos incluyen: [Menciona 3-5 temas generales, ej: Algoritmos, Tipos de Datos, Variables y Constantes, Operadores, Estructuras de Control (if, for, while), Funciones Básicas].
- Navegación sencilla entre preguntas (si aplica, o si son secuenciales).
- Indicador de respuesta correcta/incorrecta.
- Puntuación final (si la implementaste).
- Diseño adaptado para dispositivos iPhone.

2.3.4. Tecnologías Utilizadas

- **Lenguaje de Programación:** Swift
- **Entorno de Desarrollo Integrado (IDE):** Xcode
- **Gestión de Base de Datos:** Se implementó una estructura de datos interna en Swift arreglos de structs para almacenar las preguntas y respuestas. No se utilizó un motor de base de datos externo como Core Data o SQLite para esta versión, manteniendo la simplicidad del proyecto.
- **Diseño de Interfaz:** SwiftUI o UIKit.

2.4. Metodología y Fases del Desarrollo

- **Planificación e Investigación:** Definición de objetivos, alcance, selección de temas para las preguntas y diseño conceptual de la aplicación.
- **Diseño de Interfaz de Usuario (UI) y Experiencia de Usuario (UX):** Creación de wireframes y mockups básicos para la estructura visual y la navegación de la app.
- **Desarrollo y Codificación:** Implementación de la lógica de la aplicación en Swift, creación de la base de datos de preguntas y desarrollo de las vistas de usuario.
- **Pruebas y Depuración:** Realización de pruebas funcionales para identificar y corregir errores, asegurando el correcto funcionamiento del cuestionario.
- **Entrega y Documentación:** Finalización de la aplicación y elaboración del presente reporte.

Código

The screenshot shows the Xcode interface with the project 'Examen' selected. The left sidebar shows files like ExamenApp.swift, ContentView.swift, and ResultadoView.swift. The main editor window displays the code for ExamenApp.swift:

```
1 // ExamenApp.swift
2 // Examen
3 //
4 // Created by Psvel on 16/05/25.
5 //
6 import SwiftUI
7
8 @main
9 struct ExamenApp: App {
10     var body: some Scene {
11         WindowGroup {
12             ContentView()
13         }
14     }
15 }
16
17 }
```

The status bar at the bottom indicates "Line: 8 Col: 15". A note "Not Applicable" is visible in the bottom right corner.

The screenshot shows the Xcode interface with the ContentView.swift file selected. The left sidebar shows files like ExamenApp.swift, ContentView.swift, and ResultadoView.swift. The main editor window displays the code for ContentView.swift:

```
1 import SwiftUI
2
3 struct Pregunta: Identifiable {
4     let id = UUID()
5     let texto: String
6     let opciones: [String]
7     let correcta: String
8 }
9
10 let baseDePreguntas: [Pregunta] = [
11     Pregunta(texto: "¿Qué lenguaje se usa principalmente en iOS?", opciones: ["Swift", "Kotlin", "Java"], correcta: "Swift"),
12     Pregunta(texto: "¿Qué significa 'HTML'?", opciones: ["Hyper Trainer Markup Language", "Hyper Text Markup Language", "Home Tool Markup Language"], correcta: "Hyper Text Markup Language"),
13     Pregunta(texto: "¿Qué es una variable?", opciones: ["Un número fijo", "Una función", "Un espacio para almacenar datos"], correcta: "Un espacio para almacenar datos"),
14     Pregunta(texto: "¿Qué lenguaje usa el navegador para interactuar con el usuario?", opciones: ["PHP", "JavaScript", "Python"], correcta: "JavaScript"),
15     Pregunta(texto: "¿Qué significa 'CPU'?", opciones: ["Central Process Unit", "Central Processing Unit", "Computer Personal Unit"], corrects: "Central Processing Unit"),
16     Pregunta(texto: "¿Qué es el sistema operativo de una Mac?", opciones: ["Windows", "Linux", "macOS"], correcta: "macOS"),
17     Pregunta(texto: "¿Qué símbolo se usa para comentarios en Swift?", opciones: ["/", "#", "//"], correcta: "//"),
18     Pregunta(texto: "¿Qué tipo de dato representa texto?", opciones: ["String", "Int", "Bool"], correcta: "String"),
19     Pregunta(texto: "¿Qué es un bucle?", opciones: ["Una función matemática", "Una estructura para repetir código", "Un tipo de variable"], correcta: "Una estructura para repetir código"),
20     Pregunta(texto: "¿Qué hace un compilador?", opciones: ["Diseña la interfaz", "Traduce el código a lenguaje máquina", "Almacena datos"], correcta: "Traduce el código a lenguaje máquina"),
21     Pregunta(texto: "¿Qué extensión tienen los archivos de Swift?", opciones: [".java", ".swift", ".py"], correcta: ".swift"),
22     Pregunta(texto: "¿Qué es un 'array'?", opciones: ["Una imagen", "Una función", "Una lista de elementos"], correcta: "Una lista de elementos"),
23     Pregunta(texto: "¿Qué es el operador de asignación?", opciones: ["==", "=", "!="], correcta: "="),
24     Pregunta(texto: "¿Qué es una función?", opciones: ["Un número", "Una operación", "Una función matemática"], correcta: "Una función matemática")
25 ]
```

The status bar at the bottom indicates "Line: 9 Col: 1". A note "Not Applicable" is visible in the bottom right corner.

The screenshot shows the Xcode interface with the project 'Examen' open. The left sidebar shows files like ExamenApp.swift, ContentView.swift, and ResultadoView.swift. The main editor window displays the code for ResultadoView.swift:

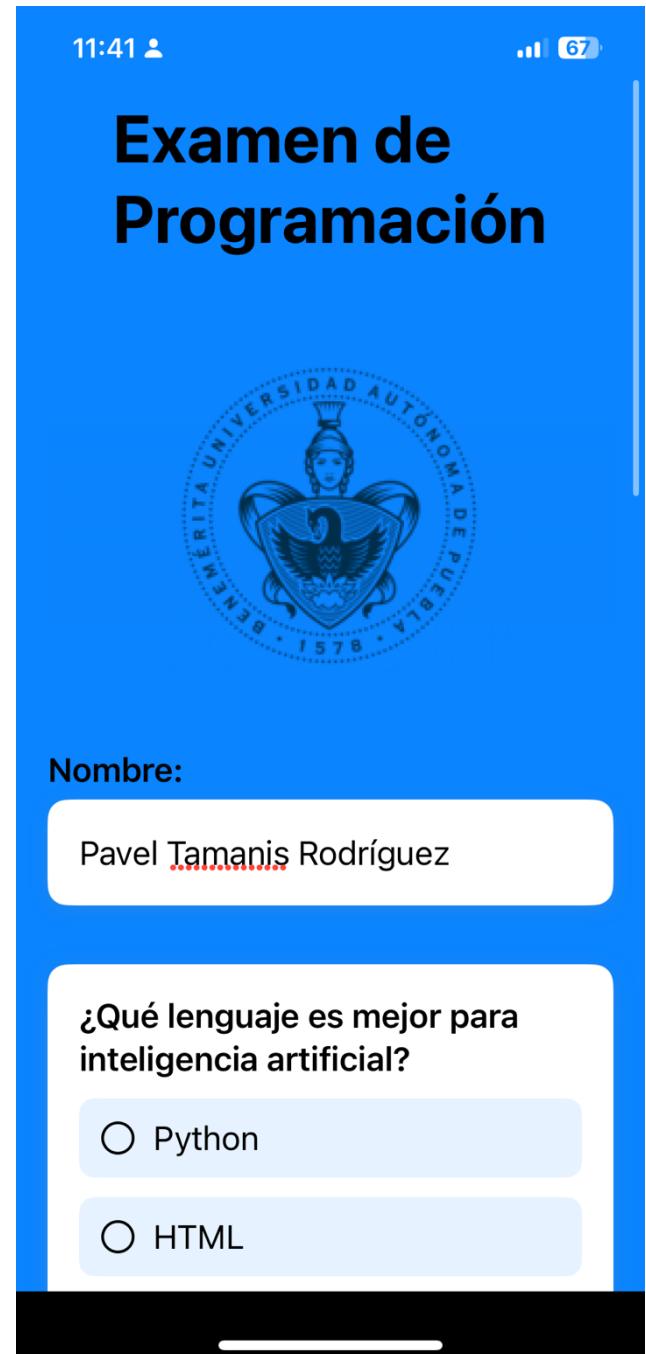
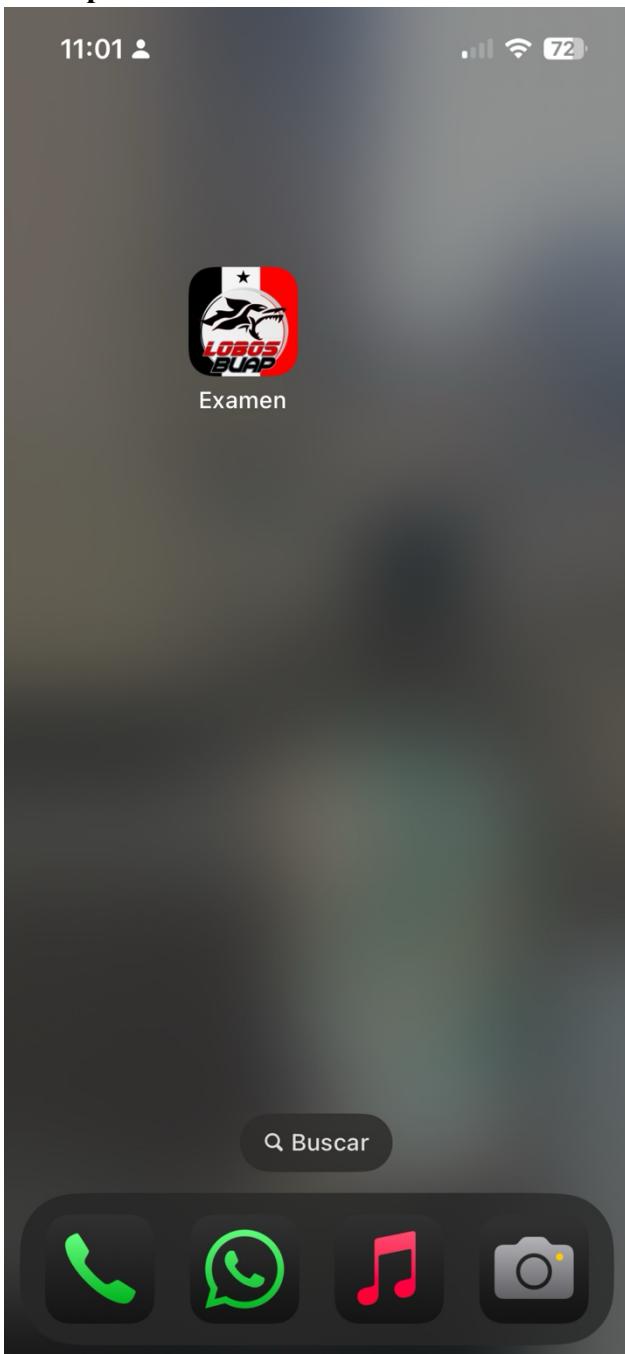
```
1 import SwiftUI
2
3 struct ResultadoView: View {
4     let preguntas: [Pregunta]
5     let respuestas: [UUID: String]
6     let nombre: String
7
8     var aciertos: Int {
9         preguntas.filter { respuestas[$0.id] == $0.correcta }.count
10    }
11
12    var puntos: Int {
13        aciertos * 2
14    }
15
16    var body: some View {
17        ScrollView {
18            VStack(alignment: .leading, spacing: 20) {
19                Text("Resultado")
20                    .font(.largeTitle)
21                    .bold()
22                    .frame(maxWidth: .infinity, alignment: .center)
23
24                Text("Alumno: \(nombre)")
25                    .font(.title2)
26
27                Text("Puntaje final: \(puntos)")
28                    .font(.title3)
29                    .bold()
30                    .padding(.bottom, 20)
31
32                // Imagen condicional
33                if puntos >= 6 {
34                    Image("ccc")
35                        .resizable()
36                        .scaledToFit()
37                        .frame(height: 300)
38                } else {
39                    Image("tictac")
40                }
41            }
42        }
43    }
44}
```

The screenshot shows the Xcode interface with the project 'Examen' open. The left sidebar shows files like ExamenApp.swift, ContentView.swift, and ResultadoView.swift. The main editor window displays the code for ResultadoView.swift, which includes additional logic for sending results via HTTP POST:

```
3 struct ResultadoView: View {
4     var body: some View {
5     }
6
7     func enviarResultados() {
8         let respuestasCodificables = respuestas.reduce(into: [String: String]()) { (dict,
9             pair) in
10            dict[pair.key.uuidString] = pair.value
11        }
12
13        let datos: [String: Any] = [
14            "nombre": nombre,
15            "respuestas": respuestasCodificables,
16            "puntaje": puntos
17        ]
18
19        guard let url = URL(string: "http://192.168.1.69:5050/guardar"),
20            let jsonData = try? JSONSerialization.data(withJSONObject: datos) else {
21            print("Error preparando JSON")
22            return
23        }
24
25        var request = URLRequest(url: url)
26        request.httpMethod = "POST"
27        request.setValue("application/json", forHTTPHeaderField: "Content-Type")
28        request.httpBody = jsonData
29
30        URLSession.shared.dataTask(with: request) { data, response, error in
31            if let error = error {
32                print("Error al enviar: \(error)")
33                return
34            }
35            if let httpResponse = response as? HTTPURLResponse {
36                print("Código de respuesta: \(httpResponse.statusCode)")
37            }
38            if let data = data, let respuesta = try? JSONSerialization.jsonObject(with: data)
39            {
40                print("Respuesta del servidor: \(respuesta)")
41            }
42        }.resume()
43    }
44}
```

Anexos

2.1. Prueba de ejecucion en Dispositivo Iphone 13 si el alumno aprobo : Te inidica respuestas correctas e incorrectas.



11:42 Nombre:

Pavel Tamanis Rodríguez

¿Qué lenguaje es mejor para inteligencia artificial?

- Python
- HTML
- CSS

¿Qué tipo de software es Word?

- Sistema operativo
- Aplicación
- Driver

¿Qué es un driver?

11:42

¿Qué es un driver?

- Controlador de hardware
- Juego
- Idioma de programación

¿Qué es una clase en programación?

- Una estructura de control
- Una plantilla para objetos
- Un tipo de bucle

¿Qué palabra se usa para declarar funciones en Swift?

- function
- func

programación?

- 11:42 66
- Una estructura de control
 - Una plantilla para objetos
 - Un tipo de bucle

¿Qué palabra se usa para declarar funciones en Swift?

- function
- func
- def

Enviar Respuestas

Autor Pavel Tamanis Rodriguez

202058576

Servicio Social FCC 2025

11:42

66

Resultado

Alumno: Pavel
Tamanis Rodríguez

Puntaje final: 8 puntos



Pregunta: ¿Qué lenguaje es mejor para inteligencia artificial?

Tu respuesta: Python

11:43

66

Pregunta: ¿Qué lenguaje es mejor para inteligencia artificial?

Tu respuesta: Python

Respuesta correcta: Python

Pregunta: ¿Qué tipo de software es Word?

Tu respuesta: Aplicación

Respuesta correcta: Aplicación

Pregunta: ¿Qué es un driver?

Tu respuesta: Controlador de hardware

Respuesta correcta: Controlador de hardware

Pregunta: ¿Qué es una clase en programación?

Tu respuesta: Una plantilla para objetos

Respuesta correcta: Una plantilla para objetos

11:43

66

Pregunta: ¿Qué tipo de software es Word?

Tu respuesta: Aplicación

Respuesta correcta: Aplicación

Pregunta: ¿Qué es un driver?

Tu respuesta: Controlador de hardware

Respuesta correcta: Controlador de hardware

Pregunta: ¿Qué es una clase en programación?

Tu respuesta: Una plantilla para objetos

Respuesta correcta: Una plantilla para objetos

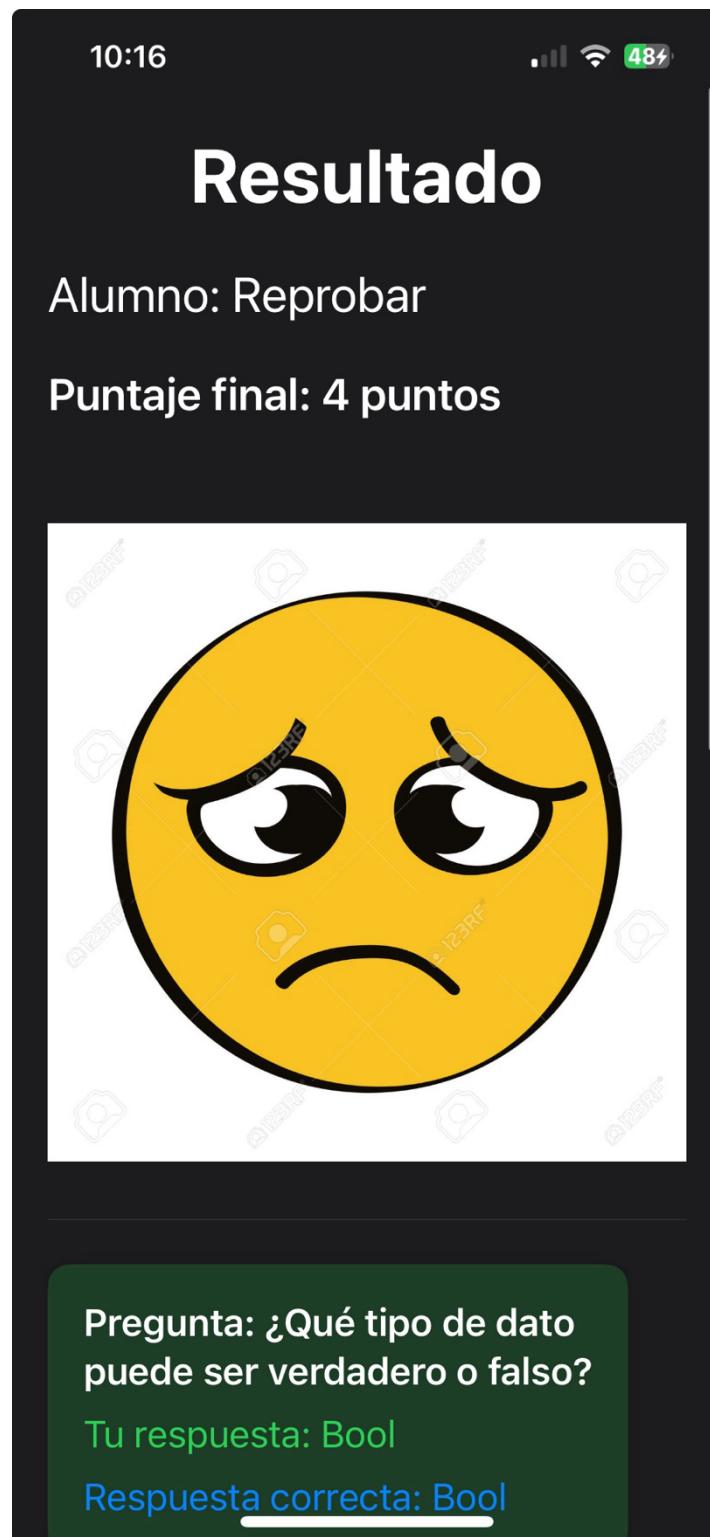
Pregunta: ¿Qué palabra se usa para declarar funciones en Swift?

Tu respuesta: def

Respuesta correcta: func

2.2. Prueba de ejecucion en Dispositivo Iphone 13 si el alumno reprobo

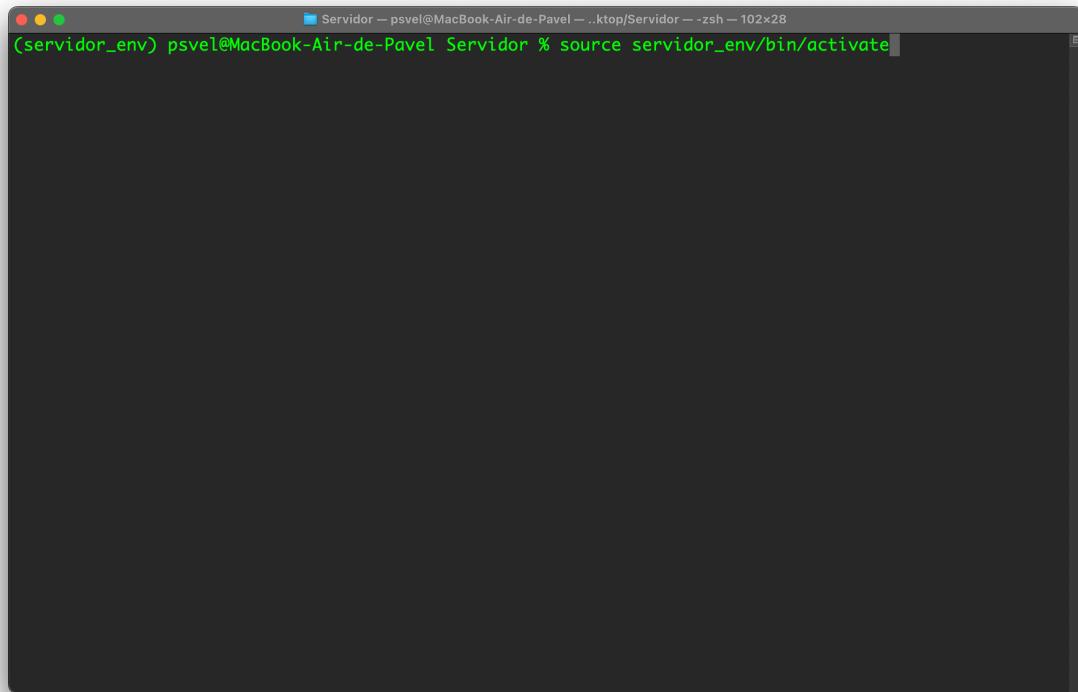
Si reprobo la imagen cambia a carita triste



2.3. Respuestas en equipo del profesor (Macbook Air)

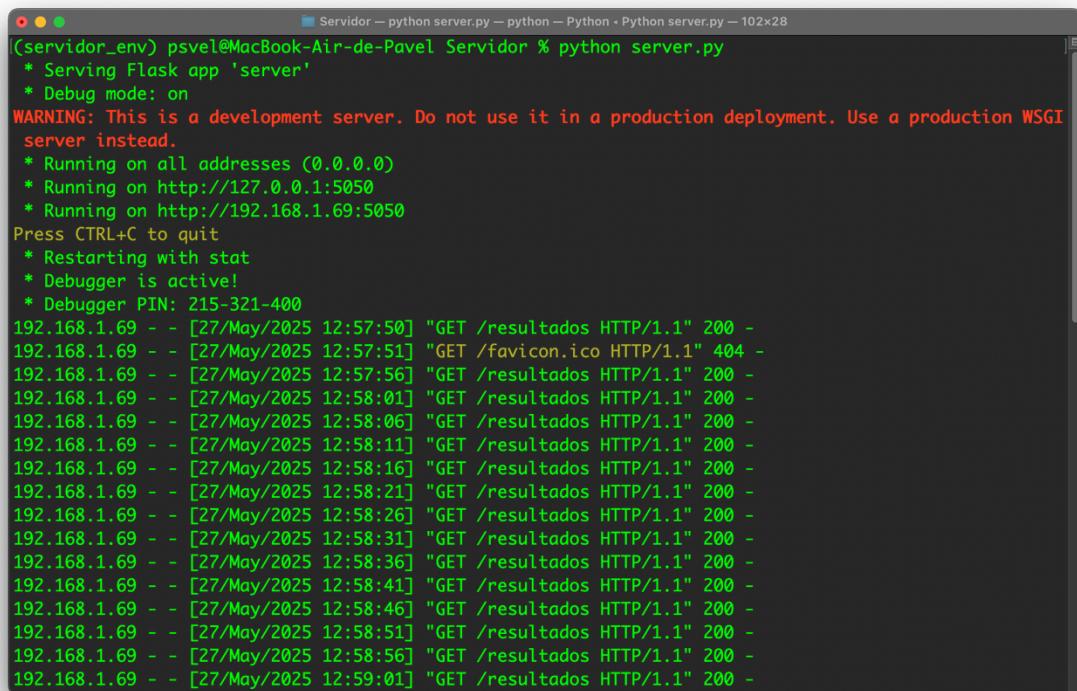
Inicio del Servidor Python

Primero, necesito poner en marcha el servidor web que gestionará los datos de los exámenes. Para ello, abro una terminal y ejecuto los siguientes comandos:



A screenshot of a terminal window titled "Servidor" running on a Mac OS X system. The window title bar shows "Servidor — psvel@MacBook-Air-de-Pavel — .ktop/Servidor — zsh — 102x28". The terminal prompt "(servidor_env) psvel@MacBook-Air-de-Pavel Servidor % source servidor_env/bin/activate" is visible at the top of the window. The rest of the window is blank black space.

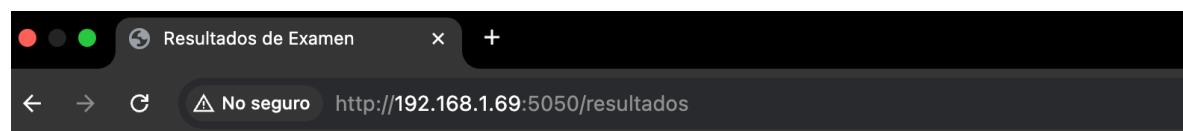
Activar el entorno virtual: Este comando prepara el ambiente específico donde tengo

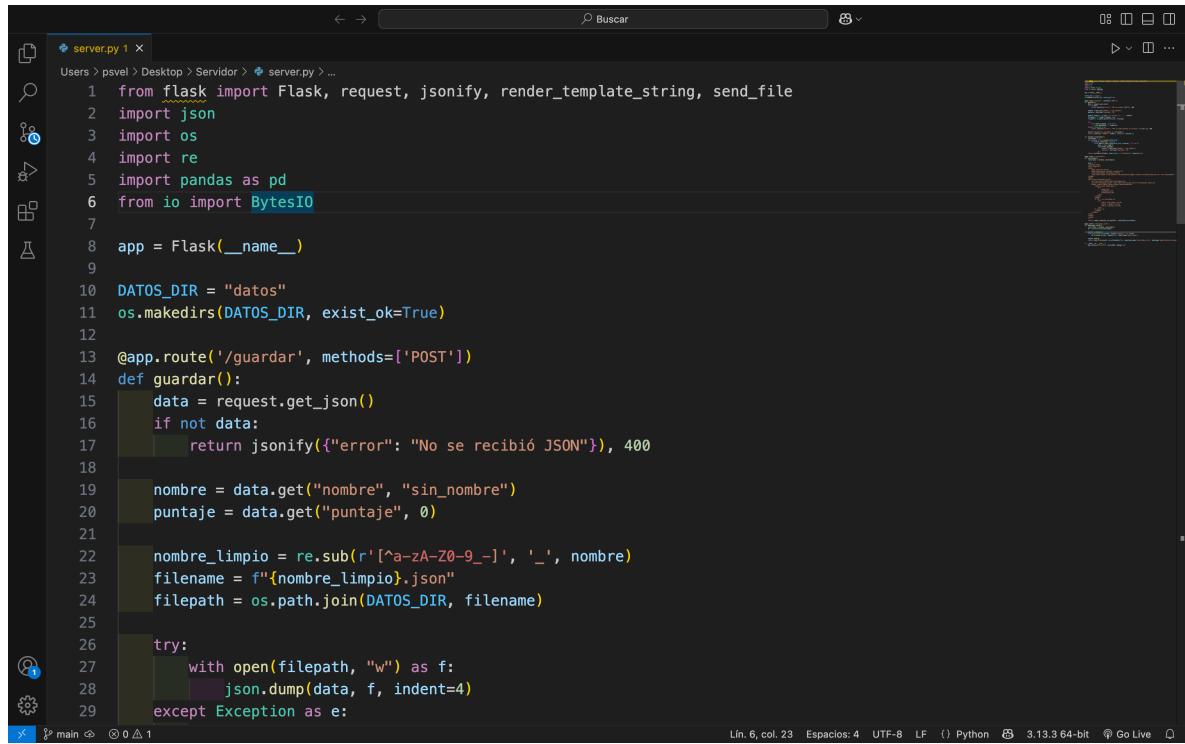


A screenshot of a terminal window titled "Servidor — python server.py — python — Python server.py — 102x28". The window shows the output of a Python script running a Flask application. It includes configuration details like port 5050 and debug mode, a warning about using it in production, and a log of multiple requests from IP 192.168.1.69 over several seconds.

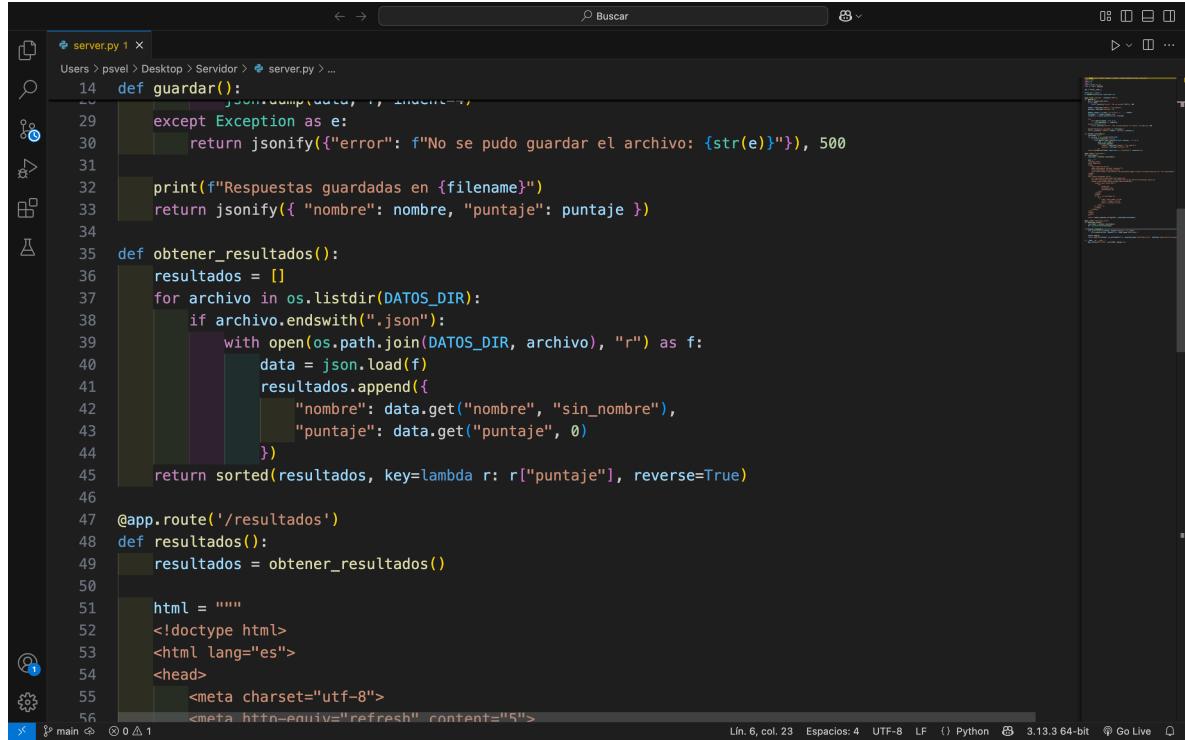
```
(servidor_env) psvel@MacBook-Air-de-Pavel Servidor % python server.py
 * Serving Flask app 'server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5050
 * Running on http://192.168.1.69:5050
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 215-321-400
192.168.1.69 - - [27/May/2025 12:57:50] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:57:51] "GET /favicon.ico HTTP/1.1" 404 -
192.168.1.69 - - [27/May/2025 12:57:56] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:01] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:06] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:11] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:16] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:21] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:26] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:31] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:36] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:41] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:46] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:51] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:58:56] "GET /resultados HTTP/1.1" 200 -
192.168.1.69 - - [27/May/2025 12:59:01] "GET /resultados HTTP/1.1" 200 -
```

Accedo con la ip de mi dispositivo y al puerto, nombre del archivo que muestra mi pagina en html.





```
server.py 1
Users > psvel > Desktop > Servidor > server.py > ...
1  from flask import Flask, request, jsonify, render_template_string, send_file
2  import json
3  import os
4  import re
5  import pandas as pd
6  from io import BytesIO
7
8  app = Flask(__name__)
9
10 DATOS_DIR = "datos"
11 os.makedirs(DATOS_DIR, exist_ok=True)
12
13 @app.route('/guardar', methods=['POST'])
14 def guardar():
15     data = request.get_json()
16     if not data:
17         return jsonify({"error": "No se recibió JSON"}), 400
18
19     nombre = data.get("nombre", "sin_nombre")
20     puntaje = data.get("puntaje", 0)
21
22     nombre_limpio = re.sub(r'^[a-zA-Z0-9_-]', '_', nombre)
23     filename = f"{nombre_limpio}.json"
24     filepath = os.path.join(DATOS_DIR, filename)
25
26     try:
27         with open(filepath, "w") as f:
28             json.dump(data, f, indent=4)
29     except Exception as e:
30         return jsonify({"error": f"No se pudo guardar el archivo: {str(e)}"}), 500
31
32     print(f"Respuestas guardadas en {filename}")
33     return jsonify({"nombre": nombre, "puntaje": puntaje})
34
35 def obtener_resultados():
36     resultados = []
37     for archivo in os.listdir(DATOS_DIR):
38         if archivo.endswith(".json"):
39             with open(os.path.join(DATOS_DIR, archivo), "r") as f:
40                 data = json.load(f)
41                 resultados.append({
42                     "nombre": data.get("nombre", "sin_nombre"),
43                     "puntaje": data.get("puntaje", 0)
44                 })
45     return sorted(resultados, key=lambda r: r["puntaje"], reverse=True)
46
47 @app.route('/resultados')
48 def resultados():
49     resultados = obtener_resultados()
50
51     html = """
52     <!doctype html>
53     <html lang="es">
54         <head>
55             <meta charset="utf-8">
56             <meta http-equiv="refresh" content="5">
57     </head>
58     <body>
59         <h1>Resultados</h1>
60         <table border="1">
61             <thead>
62                 <tr>
63                     <th>Nombre</th>
64                     <th>Puntaje</th>
65                 </tr>
66             </thead>
67             <tbody>
68                 <tr>
69                     <td>{{ resultado.nombre }}</td>
70                     <td>{{ resultado.puntaje }}</td>
71                 </tr>
72             </tbody>
73         </table>
74     </body>
75 </html>
76 """
77     return render_template_string(html)
```



```
server.py 1
Users > psvel > Desktop > Servidor > server.py > ...
14 def guardar():
15     json.dump(data, f, indent=4)
16     except Exception as e:
17         return jsonify({"error": f"No se pudo guardar el archivo: {str(e)}"}), 500
18
19 print(f"Respuestas guardadas en {filename}")
20 return jsonify({"nombre": nombre, "puntaje": puntaje})
21
22 def obtener_resultados():
23     resultados = []
24     for archivo in os.listdir(DATOS_DIR):
25         if archivo.endswith(".json"):
26             with open(os.path.join(DATOS_DIR, archivo), "r") as f:
27                 data = json.load(f)
28                 resultados.append({
29                     "nombre": data.get("nombre", "sin_nombre"),
30                     "puntaje": data.get("puntaje", 0)
31                 })
32     return sorted(resultados, key=lambda r: r["puntaje"], reverse=True)
33
34 @app.route('/resultados')
35 def resultados():
36     resultados = obtener_resultados()
37
38     html = """
39     <!doctype html>
40     <html lang="es">
41         <head>
42             <meta charset="utf-8">
43             <meta http-equiv="refresh" content="5">
44         </head>
45         <body>
46             <h1>Resultados</h1>
47             <table border="1">
48                 <thead>
49                     <tr>
50                         <th>Nombre</th>
51                         <th>Puntaje</th>
52                     </tr>
53                 </thead>
54                 <tbody>
55                     <tr>
56                         <td>{{ resultado.nombre }}</td>
57                         <td>{{ resultado.puntaje }}</td>
58                     </tr>
59                 </tbody>
60             </table>
61         </body>
62     </html>
63 """
64     return render_template_string(html)
```

server.py

```
48 def resultados():
51     html = """
52     <!doctype html>
53     <html lang="es">
54         <head>
55             <meta charset="utf-8">
56             <meta http-equiv="refresh" content="5">
57             <title>Resultados de Examen</title>
58             <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
59         </head>
60         <body>
61             <div class="container my-4">
62                 <h1 class="mb-4">Resultados de Examen</h1>
63                 <a href="/descargar_excel" class="btn btn-success mb-3">⬇ Descargar Excel</a>
64                 <table class="table table-striped table-bordered">
65                     <thead class="table-dark">
66                         <tr>
67                             <th>#</th>
68                             <th>Nombre</th>
69                             <th>Puntaje</th>
70                         </tr>
71                     </thead>
72                     <tbody>
73                         {% for r in resultados %}
74                             <tr>
75                                 <td>{{ loop.index }}</td>
76                                 <td>{{ r.nombre }}</td>
77                                 <td>{{ r.puntaje }}</td>
78                         </tr>
79                         {% endfor %}
80                     </tbody>
81                 </table>
82             </div>
83         </body>
84     </html>
85     """
86     return render_template_string(html, resultados=resultados)
87
88 @app.route('/descargar_excel')
89 def descargar_excel():
90     resultados = obtener_resultados()
91     df = pd.DataFrame(resultados)
92
93     output = BytesIO()
94     with pd.ExcelWriter(output, engine='openpyxl') as writer:
95         df.to_excel(writer, index=False, sheet_name='Resultados')
96
97     output.seek(0)
98     return send_file(output, as_attachment=True, download_name="resultados.xlsx", mimetype='application/vnd.ms-excel')
99
100 if __name__ == '__main__':
101     app.run(host='0.0.0.0', port=5050, debug=True)
```

server.py

```
48 def resultados():
51     html = """
52     <!doctype html>
53     <html lang="es">
54         <head>
55             <meta charset="utf-8">
56             <meta http-equiv="refresh" content="5">
57             <title>Resultados de Examen</title>
58             <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
59         </head>
60         <body>
61             <div class="container my-4">
62                 <h1 class="mb-4">Resultados de Examen</h1>
63                 <a href="/descargar_excel" class="btn btn-success mb-3">⬇ Descargar Excel</a>
64                 <table class="table table-striped table-bordered">
65                     <thead class="table-dark">
66                         <tr>
67                             <th>#</th>
68                             <th>Nombre</th>
69                             <th>Puntaje</th>
70                         </tr>
71                     </thead>
72                     <tbody>
73                         {% for r in resultados %}
74                             <tr>
75                                 <td>{{ loop.index }}</td>
76                                 <td>{{ r.nombre }}</td>
77                                 <td>{{ r.puntaje }}</td>
78                         </tr>
79                         {% endfor %}
80                     </tbody>
81                 </table>
82             </div>
83         </body>
84     </html>
85     """
86     return render_template_string(html, resultados=resultados)
87
88 @app.route('/descargar_excel')
89 def descargar_excel():
90     resultados = obtener_resultados()
91     df = pd.DataFrame(resultados)
92
93     output = BytesIO()
94     with pd.ExcelWriter(output, engine='openpyxl') as writer:
95         df.to_excel(writer, index=False, sheet_name='Resultados')
96
97     output.seek(0)
98     return send_file(output, as_attachment=True, download_name="resultados.xlsx", mimetype='application/vnd.ms-excel')
99
100 if __name__ == '__main__':
101     app.run(host='0.0.0.0', port=5050, debug=True)
```

Resultados de Examen

[Descargar Excel](#)

#	Nombre	Puntaje
1	José Lobato	10
2	Juan Carlos	10
3	Pera	10
4	Perra	8
5	Pavel Tamanis Rodríguez	8
6	Pifa	8
7	VALERIA JUÁREZ	8
8	Por	6
9	Mm	6
10	José Pérez	6
11	Hay	6
12	José José	6
13	A	4
14	Roman	4
15	Atípica	4

Excel

Conclusión

La realización del Servicio Social mediante el desarrollo de la aplicación móvil educativa “Examen” ha representado una valiosa oportunidad para aplicar y expandir los conocimientos adquiridos durante mi formación en Ingeniería en Ciencias de la Computación.

A lo largo de estos 5 años en la facultad he aprendido varios conceptos básicos de programación.

Logré desarrollar una aplicación que busca facilitar el aprendizaje de los fundamentos de programación también obtuve importantes aprendizajes tanto a nivel técnico como personal. Crecí como persona y como profesional durante este tiempo.

Agradezco a mi profesor **Alberto Román Flores** por todo su tiempo, dedicación, esfuerzo y paciencia que tuvo conmigo durante el servicio y todas las clases que me impartió. Me voy feliz por todo el conocimiento y el tiempo que pase en esta Universidad y ser parte de la poderosa Facultad de Ciencias de la Computación. Espero dar lo mejor de mi hoy y siempre.

Hoja de Firmas

Hoja de Firmas de Servicio Social

Nombre Completo: Pavel Tamanis Rodriguez

Matrícula : 202058576

Carrera o Programa Académico: Ingenieria en Ciencias de la Computación

Periodo que Cubre la Hoja: Fecha del 06/01/2025 al 06/07/2025

Actividad Realizada: Desarrollo de una aplicación móvil para generar un examen
(207176)

Total de Horas Cubiertas en ese Periodo: 480 horas



Atentamente

M.C ALBERTO ROMAN FLORES

COORDINADOR DE LABORATORIO



Atentamente

PAVEL TAMANIS RODRIGUEZ

ALUMNO