Universidad Tecnologica de Tijuana

Desarrollo de Software Multiplataforma

Subject:

Desarrollo móvil multiplataforma

Teacher:

Ray Brunett Parra Galaviz

Students:

Alvarez Galindo Aldo Yamil

Gómez Miramontes Daniel

Mayo Ramos Ángel David

Muñoz Reynoso Oscar Gael

Introduction:

The Workbit system will be deployed in designated study rooms, reading areas, and small classrooms. Its functionality supports environmental monitoring, attendance tracking, room readiness automation, and analytics dashboards for resource optimization.

Description of the problem.

Environments with Poor Air Quality

Many classrooms and workspaces lack adequate ventilation.

High $CO_2$ levels affect concentration, performance, and health.

There are no alerts or indicators visible to users or administrators.

Lack of Automation in Space Management

The use of cubicles or rooms is managed manually, without precise control.

There is no automated record of who enters or how long they stay.

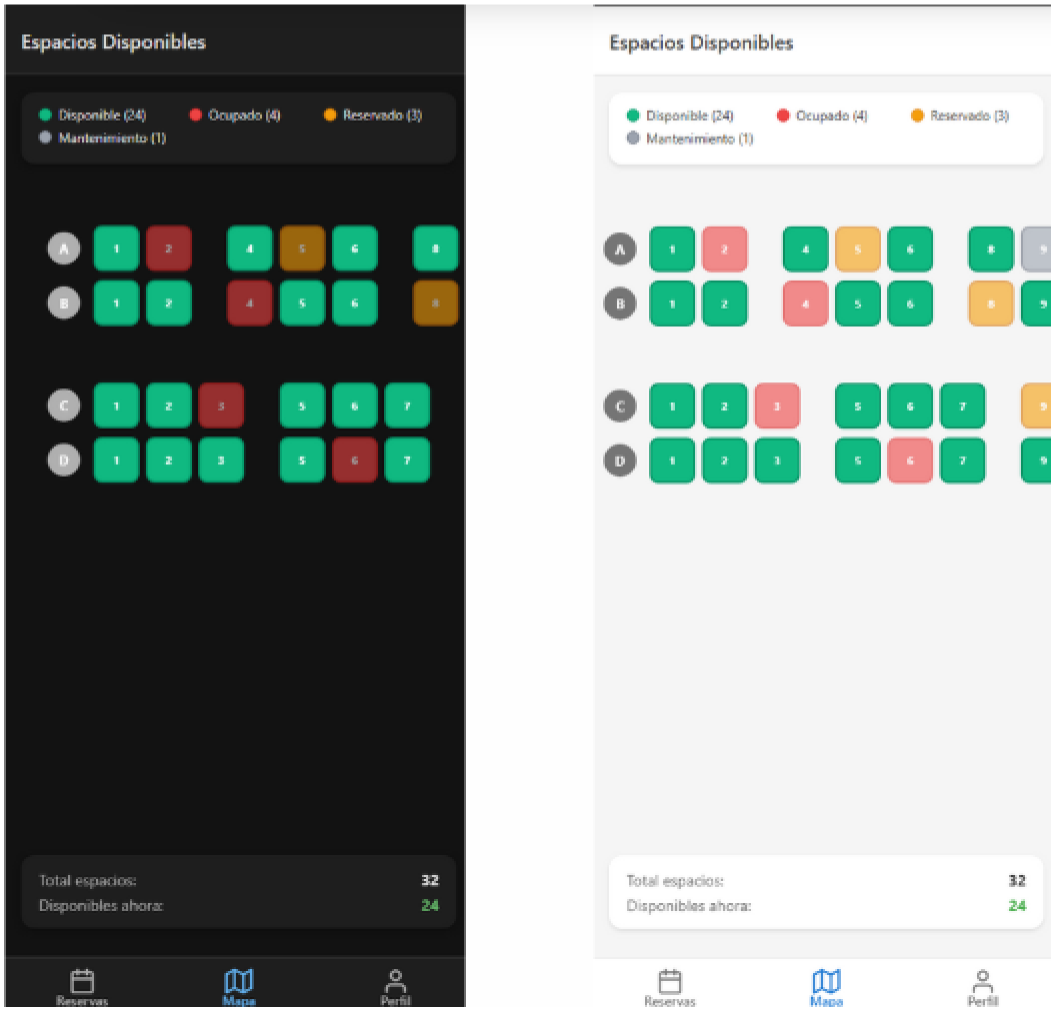
## General objetive

Develop an intelligent workspace management system that, using IoT technologies and role-based digital access, automates environmental control, records space usage, and facilitates real-time monitoring from web and mobile platforms.

Progress in front-end mobile app

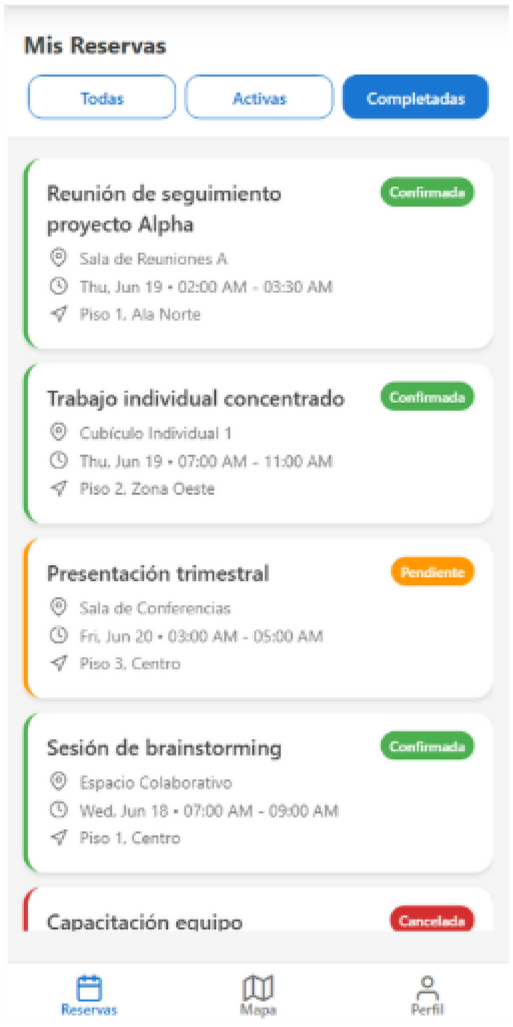
1. section to view the spaces

   In this section we can visualize the spaces that are available, those in green are those that are available for reservation, those in red are those that are currently being occupied and finally, those in orange are those that are already reserved.



2. booking history

   On the history side we can visualize all the reservations we have made and filter them by the reservations that have already been completed, by those that are pending or scheduled and by those that are active.



Mis Reservas

Reunión de seguimiento proyecto Alpha — Confirmada
Sala de Reuniones A
Thu, Jun 19 • 02:00 AM - 03:30 AM
Piso 1, Ala Norte

Trabajo individual concentrado — Confirmada
Cubículo Individual 1
Thu, Jun 19 • 07:00 AM - 11:00 AM
Piso 2, Zona Oeste

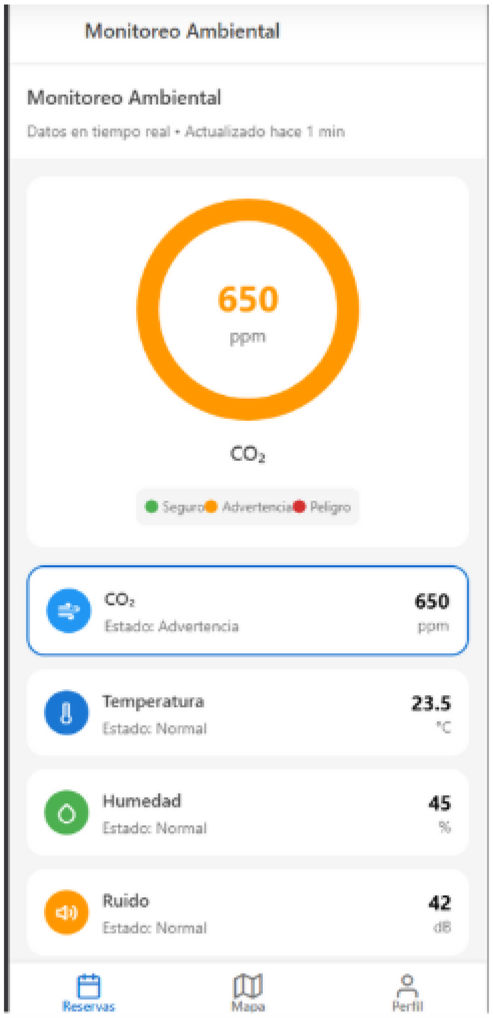
Presentación trimestral — Pendiente
Sala de Conferencias
Fri, Jun 20 • 03:00 AM - 05:00 AM
Piso 3, Centro

Sesión de brainstorming — Confirmada
Espacio Colaborativo
Wed, Jun 18 • 07:00 AM - 09:00 AM
Piso 1, Centro

Capacitación equipo — Cancelada

3. environmental monitoring

Once the user has entered his space, he can visualize its conditions, such as CO2, temperature, humidity and even noise.


Monitoreo Ambiental

Monitoreo Ambiental

Datos en tiempo real • Actualizado hace 1 min

650 ppm

$CO_2$

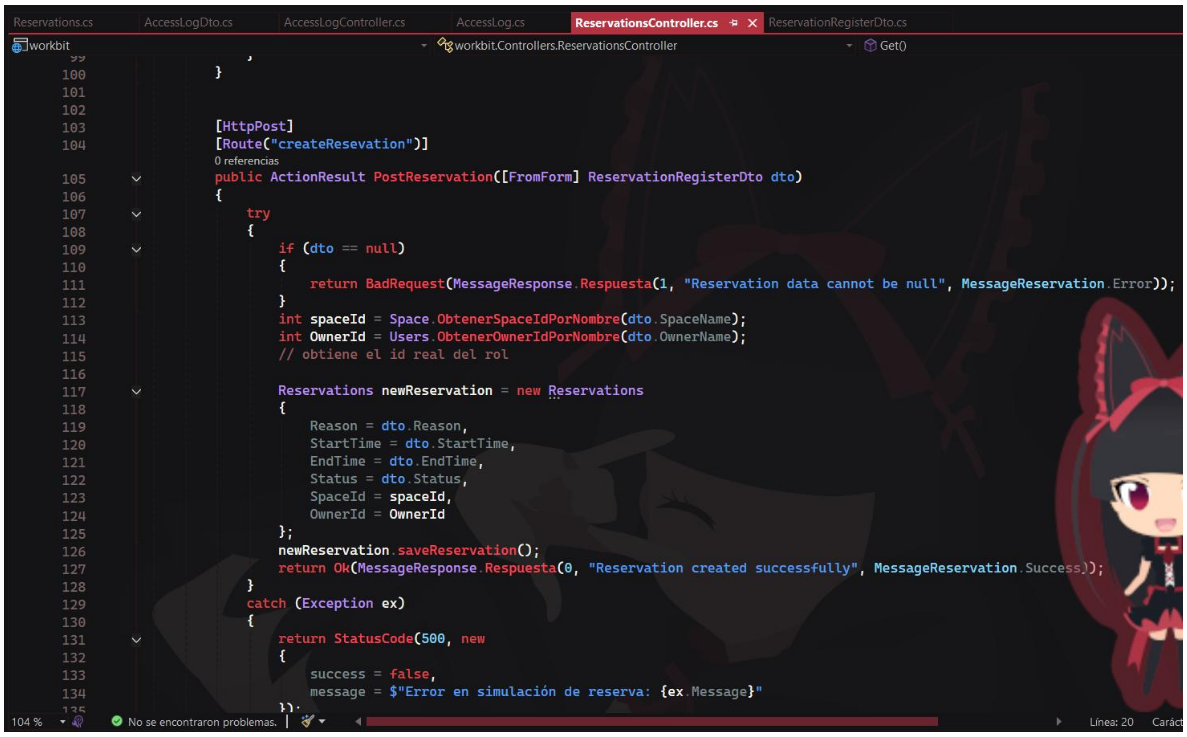
● Seguro ● Advertencia ● Peligro

| | | |
|---|---|---|
| $CO_2$ | Estado: Advertencia | 650 ppm |
| Temperatura | Estado: Normal | 23.5 °C |
| Humedad | Estado: Normal | 45 % |
| Ruido | Estado: Normal | 42 dB |

Progress in back-end mobile app

4. code for reservations

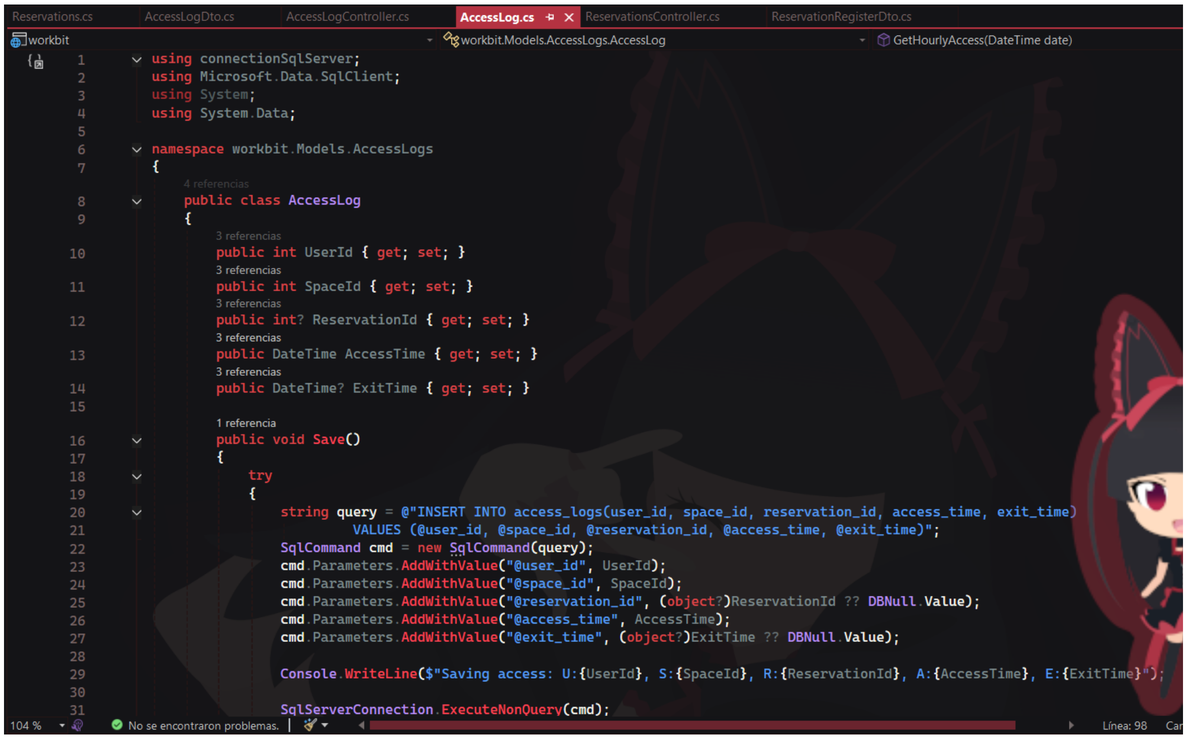
   The following code shows the reservation system controller, which receives the data needed to make a reservation and executes an insert procedure in the database.


Reservations.cs
AccessLogDto.cs
AccessLogController.cs
AccessLog.cs
ReservationsController.cs
ReservationRegisterDto.cs
workbit
workbit.Controllers.ReservationsController
Get()

```
        }


        [HttpPost]
        [Route("createResevation")]
        0 referencias
        public ActionResult PostReservation([FromForm] ReservationRegisterDto dto)
        {
            try
            {
                if (dto == null)
                {
                    return BadRequest(MessageResponse.Respuesta(1, "Reservation data cannot be null", MessageReservation.Error));
                }
                int spaceId = Space.ObtenerSpaceIdPorNombre(dto.SpaceName);
                int OwnerId = Users.ObtenerOwnerIdPorNombre(dto.OwnerName);
                // obtiene el id real del rol

                Reservations newReservation = new Reservations
                {
                    Reason = dto.Reason,
                    StartTime = dto.StartTime,
                    EndTime = dto.EndTime,
                    Status = dto.Status,
                    SpaceId = spaceId,
                    OwnerId = OwnerId
                };
                newReservation.saveReservation();
                return Ok(MessageResponse.Respuesta(0, "Reservation created successfully", MessageReservation.Success));
            }
            catch (Exception ex)
            {
                return StatusCode(500, new
                {
                    success = false,
                    message = $"Error en simulación de reserva: {ex.Message}"
```

5. code for access control

The following code shows the functions to insert a new record into the database, which is a new access to a workspace.

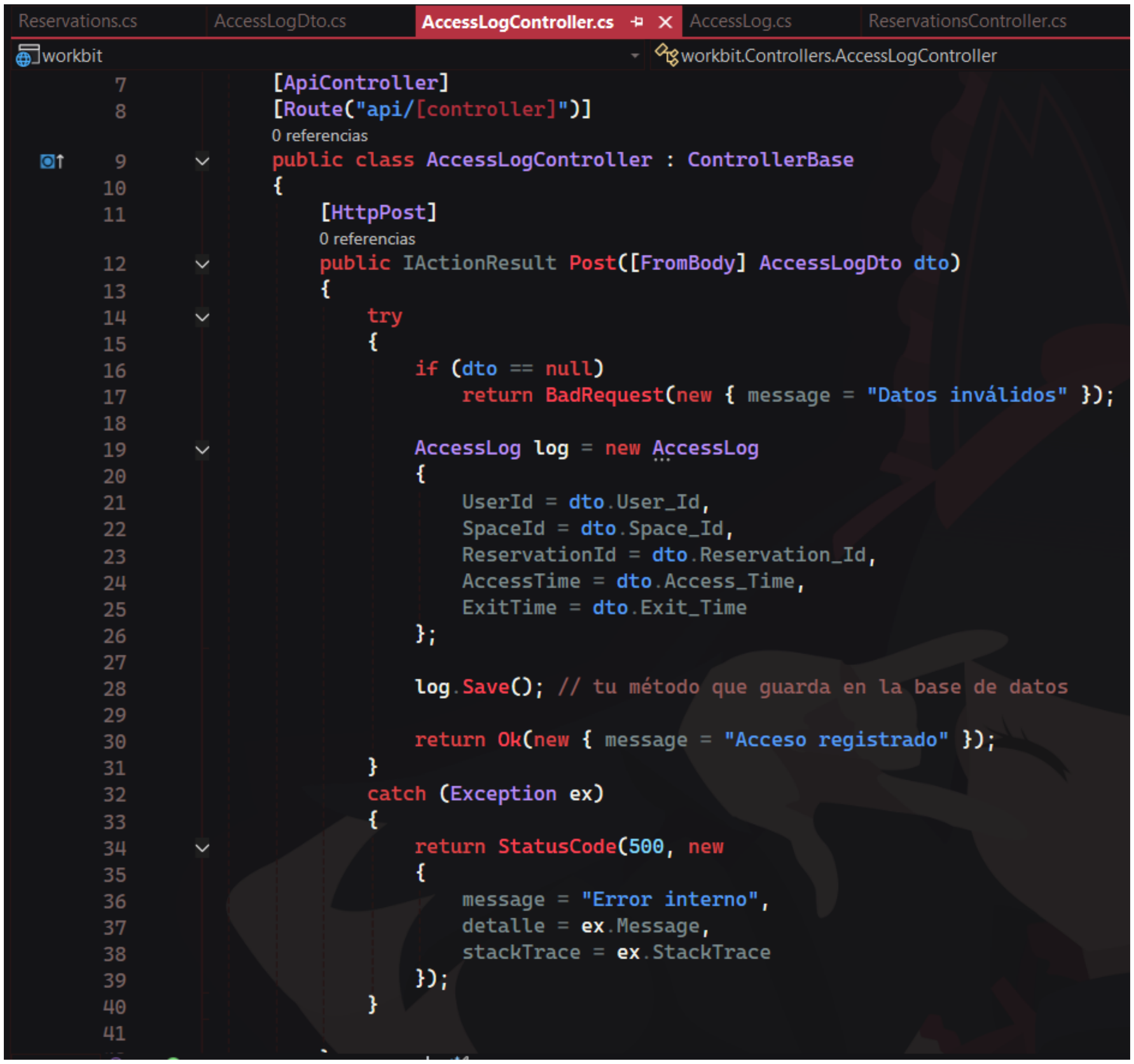


Reservations.cs | AccessLogDto.cs | AccessLogController.cs | AccessLog.cs | ReservationsController.cs | ReservationRegisterDto.cs

workbit | workbit.Models.AccessLogs.AccessLog | GetHourlyAccess(DateTime date)

```csharp
using connectionSqlServer;
using Microsoft.Data.SqlClient;
using System;
using System.Data;

namespace workbit.Models.AccessLogs
{
    4 referencias
    public class AccessLog
    {
        3 referencias
        public int UserId { get; set; }
        3 referencias
        public int SpaceId { get; set; }
        3 referencias
        public int? ReservationId { get; set; }
        3 referencias
        public DateTime AccessTime { get; set; }
        3 referencias
        public DateTime? ExitTime { get; set; }

        1 referencia
        public void Save()
        {
            try
            {
                string query = @"INSERT INTO access_logs(user_id, space_id, reservation_id, access_time, exit_time)
                         VALUES (@user_id, @space_id, @reservation_id, @access_time, @exit_time)";
                SqlCommand cmd = new SqlCommand(query);
                cmd.Parameters.AddWithValue("@user_id", UserId);
                cmd.Parameters.AddWithValue("@space_id", SpaceId);
                cmd.Parameters.AddWithValue("@reservation_id", (object?)ReservationId ?? DBNull.Value);
                cmd.Parameters.AddWithValue("@access_time", AccessTime);
                cmd.Parameters.AddWithValue("@exit_time", (object?)ExitTime ?? DBNull.Value);

                Console.WriteLine($"Saving access: U:{UserId}, S:{SpaceId}, R:{ReservationId}, A:{AccessTime}, E:{ExitTime}");

                SqlServerConnection.ExecuteNonQuery(cmd);
```

104 % | No se encontraron problemas. | Línea: 98 | Car



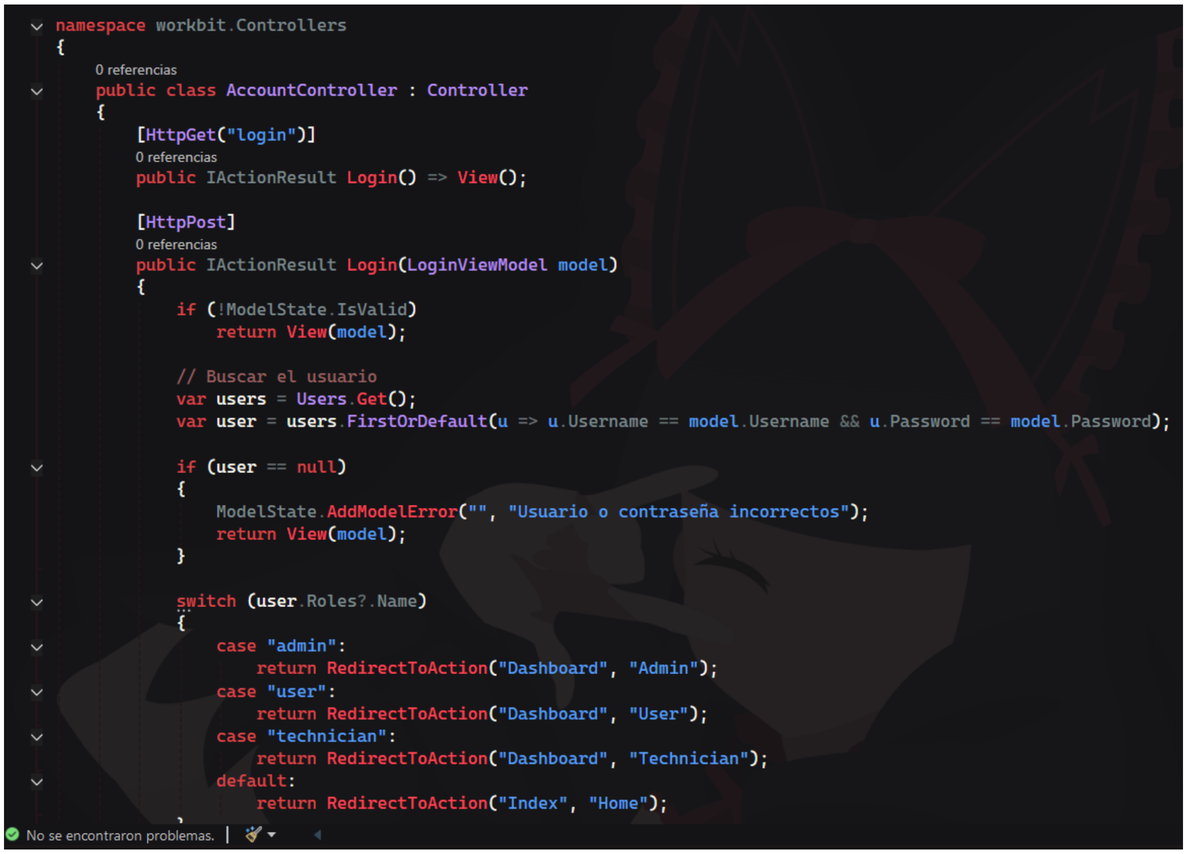
Reservations.cs | AccessLogDto.cs | AccessLogController.cs | AccessLog.cs | ReservationsController.cs

workbit | workbit.Controllers.AccessLogController

```csharp
[ApiController]
[Route("api/[controller]")]
0 referencias
public class AccessLogController : ControllerBase
{
    [HttpPost]
    0 referencias
    public IActionResult Post([FromBody] AccessLogDto dto)
    {
        try
        {
            if (dto == null)
                return BadRequest(new { message = "Datos inválidos" });

            AccessLog log = new AccessLog
            {
                UserId = dto.User_Id,
                SpaceId = dto.Space_Id,
                ReservationId = dto.Reservation_Id,
                AccessTime = dto.Access_Time,
                ExitTime = dto.Exit_Time
            };

            log.Save(); // tu método que guarda en la base de datos

            return Ok(new { message = "Acceso registrado" });
        }
        catch (Exception ex)
        {
            return StatusCode(500, new
            {
                message = "Error interno",
                detalle = ex.Message,
                stackTrace = ex.StackTrace
            });
        }

```

## 6. login code

In the following code, you manage which dashboard each user is sent to according to their role.

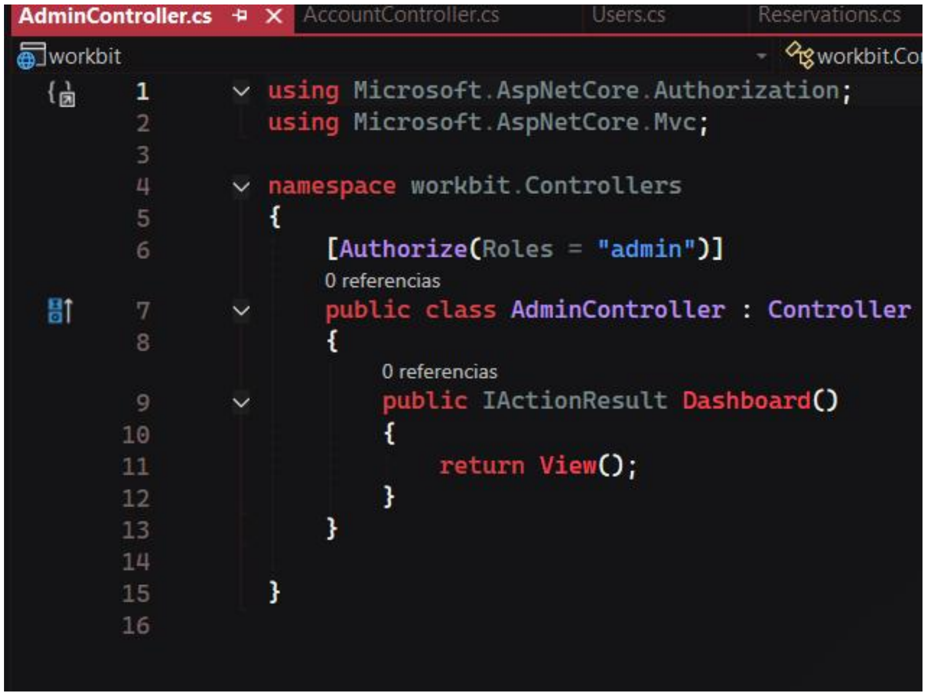


```
namespace workbit.Controllers
{
    0 referencias
    public class AccountController : Controller
    {
        [HttpGet("login")]
        0 referencias
        public IActionResult Login() => View();

        [HttpPost]
        0 referencias
        public IActionResult Login(LoginViewModel model)
        {
            if (!ModelState.IsValid)
                return View(model);

            // Buscar el usuario
            var users = Users.Get();
            var user = users.FirstOrDefault(u => u.Username == model.Username && u.Password == model.Password);

            if (user == null)
            {
                ModelState.AddModelError("", "Usuario o contraseña incorrectos");
                return View(model);
            }

            switch (user.Roles?.Name)
            {
                case "admin":
                    return RedirectToAction("Dashboard", "Admin");
                case "user":
                    return RedirectToAction("Dashboard", "User");
                case "technician":
                    return RedirectToAction("Dashboard", "Technician");
                default:
                    return RedirectToAction("Index", "Home");
```

No se encontraron problemas.

7. code for role control

The following code is for each user's dashboard according to their role.

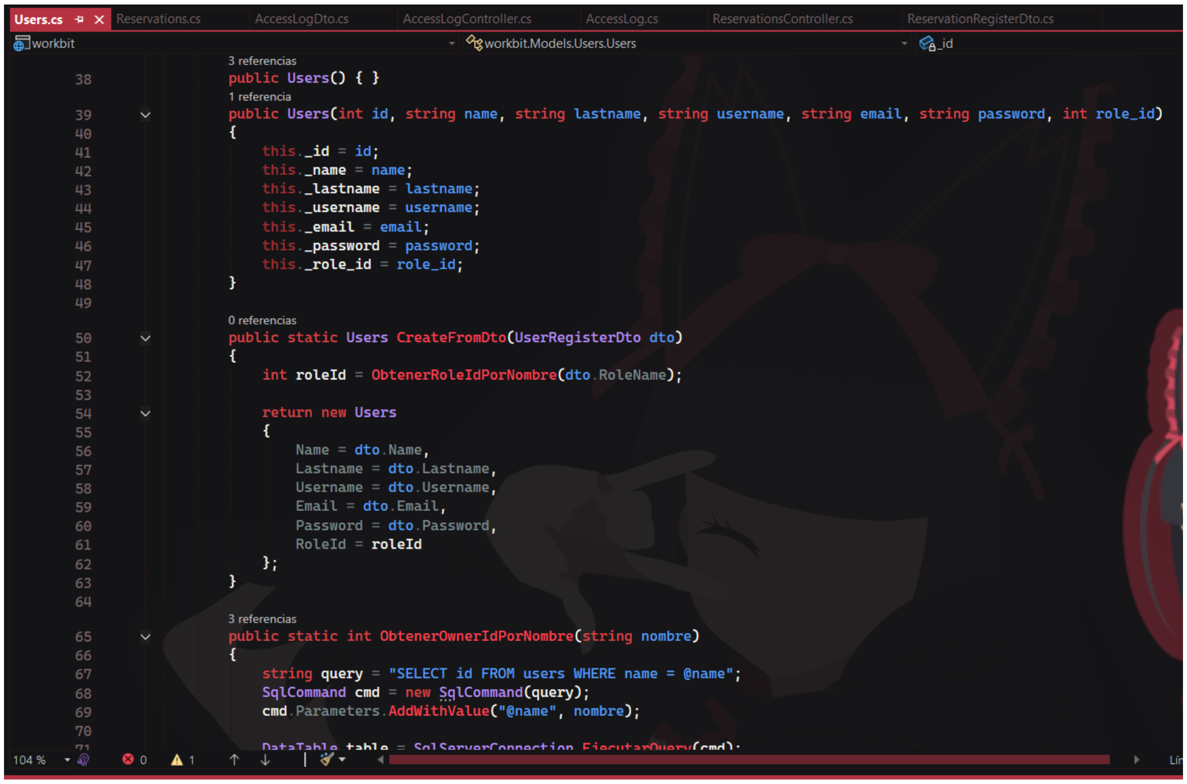

AdminController.cs AccountController.cs Users.cs Reservations.cs

workbit workbit.Co

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace workbit.Controllers
{
    [Authorize(Roles = "admin")]
    0 referencias
    public class AdminController : Controller
    {
        0 referencias
        public IActionResult Dashboard()
        {
            return View();
        }
    }

}
```

8. code for user functionalities

In the following code, perform simple actions with the user's data, such as registering them or displaying their data to update them.



```
        3 referencias
        public Users() { }
        1 referencia
        public Users(int id, string name, string lastname, string username, string email, string password, int role_id)
        {
            this._id = id;
            this._name = name;
            this._lastname = lastname;
            this._username = username;
            this._email = email;
            this._password = password;
            this._role_id = role_id;
        }

        0 referencias
        public static Users CreateFromDto(UserRegisterDto dto)
        {
            int roleId = ObtenerRoleIdPorNombre(dto.RoleName);

            return new Users
            {
                Name = dto.Name,
                Lastname = dto.Lastname,
                Username = dto.Username,
                Email = dto.Email,
                Password = dto.Password,
                RoleId = roleId
            };
        }

        3 referencias
        public static int ObtenerOwnerIdPorNombre(string nombre)
        {
            string query = "SELECT id FROM users WHERE name = @name";
            SqlCommand cmd = new SqlCommand(query);
            cmd.Parameters.AddWithValue("@name", nombre);

            DataTable table = SqlServerConnection.EjecutarQuery(cmd);
```

## Coclusion

This first preview shows the development of the main functions of the mobile app already working and operating without any errors. At this point, all that remains is to finish the minor functionalities and complete the front-end development.