# 1 – Tablas



# 2 - Repositorio

Creamos el repositorio en GitHub y lo llamamos **Vehicles**.

Debe tener el gitignore de VisualStudio.

# 3 - Solución

Copiamos la ruta del repositorio y vamos a Visual Studio y clonamos el Repositorio en la Carpeta **F:\Xamarin/Vehicles**

Dentro de la Solución **Vehicles** creamos los Proyectos:

* Biblioteca de Clases (.NET Framework) en .NET 5, y lo llamamos **Vehicles.Common**
* Aplicación web de ASP.NET Core (modelo Vista-Controlador) en .NET 5, y lo llamamos **Vehicles.Api**

# 4 - Creación de la Base de Datos

En **Vehicles.Api** creamos la carpeta **Data** y dentro creamos la carpeta **Entities**

Dentro creamos la Clase **VehicleType**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class VehicleType

{

public int Id { get; set; }

[Display(Name = " Tipo de vehículo")]

[MaxLength(50, ErrorMessage ="El campo {0} no puede tener más de {1} caracteres")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public string Description { get; set; }

}

}

Dentro de la carpeta **Data** creamos la Clase **DataContext**

using Microsoft.EntityFrameworkCore;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Data

{

public class DataContext : DbContext

{

public DataContext(DbContextOptions<DataContext> options) : base(options)

{

}

public DbSet<VehicleType> VehicleTypes { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<VehicleType>().HasIndex(x => x.Description).IsUnique();

}

}

}

Configuramos la conexión a la Base de Datos en el archivo **appsettings.json**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": { "DefaultConnection": "Server=keypress.serveftp.net;Database=LuisVehicles;User Id=sa;password=sentey14$;Trusted\_Connection=False;MultipleActiveResultSets=true" }

}

En el proyecto **Vehicles.Api** agregamos los Nuggets:

* Microsoft.EntityFrameworkCore.SqlServer
* Microsoft.EntityFrameworkCore.Tools

En el archivo **Startup.cs** hacemos:

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddDbContext<DataContext>(x =>

{

x.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

});

}

Vamos a la **Consola del Administrador de paquetes**.

Nos aseguramos que diga **Vehicles.Api** arriba y abajo

Ejecutamos los comandos:

**add-migration AddTableVehicleType**

**update-database**

# 5 - Vehicle Types

Creamos el controlador **VehicleTypesController** usando



**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente**

En la Vista **\_Layout.cshtml** hacemos:

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

@\*<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>\*@

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="VehicleTypes" **asp-action**="Index">Tipos de Vehículo</a>

</li>

<li class="nav-item">

@\*<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a\*@>

</li>

</ul>

</div>

Cambiamos la vista **Index** de **VehicleTypes**

@model IEnumerable<Vehicles.Api.Data.Entities.VehicleType>

@{

ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

<a **asp-action**="Create" class="btn btn-primary">Crear Nuevo Vehículo</a>

</p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Tipos de Vehiculo</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Description)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Description)

</td>

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<a **asp-action**="Delete" **asp-route-id**="@item.Id" class="btn btn-danger">Borrar</a>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

});

</script>

}

Cambiamos la vista **Create** de **VehicleTypes**

@model Vehicles.Api.Data.Entities.VehicleType

@{

ViewData["Title"] = "Create";

}

<h1>Crear</h1>

<h4>Tipo de Vehículo</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Create">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label **asp-for**="Description" class="control-label"></label>

<input **asp-for**="Description" class="form-control" />

<span **asp-validation-for**="Description" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Crear" class="btn btn-primary" />

<a **asp-action**="Index"class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Cambiamos la vista **Edit** de **VehicleTypes**

@model Vehicles.Api.Data.Entities.VehicleType

@{

ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Tipo de Vehículo</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Edit">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<div class="form-group">

<label **asp-for**="Description" class="control-label"></label>

<input **asp-for**="Description" class="form-control" />

<span **asp-validation-for**="Description" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

<div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Borramos la acción Details del Controlador y también borramos la Vista Details

Cambiamos el método Delete en el Controlador:

// GET: VehicleTypes/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var vehicleType = await \_context.VehiclesTypes

.FirstOrDefaultAsync(m => m.Id == id);

if (vehicleType == null)

{

return NotFound();

}

\_context.VehiclesTypes.Remove(vehicleType);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

Borramos el Delete Post y borramos la Vista Delete.

En la carpeta **wwwroot/js** creamos el archivo de JavaScript que lo llamamos **deleteDialog**

(function (soccerDeleteDialog) {

var methods = {

"openModal": openModal,

"deleteItem": deleteItem

};

var item\_to\_delete;

function openModal(modalName, classOrId, sourceEvent, deletePath, eventClassOrId) {

var textEvent;

if (classOrId) {

textEvent = "." + modalName;

} else {

textEvent = "#" + modalName;

}

$(textEvent).click((e) => {

item\_to\_delete = e.currentTarget.dataset.id;

deleteItem(sourceEvent, deletePath, eventClassOrId);

});

}

function deleteItem(sourceEvent, deletePath, eventClassOrId) {

var textEvent;

if (eventClassOrId) {

textEvent = "." + sourceEvent;

} else {

textEvent = "#" + sourceEvent;

}

$(textEvent).click(function () {

window.location.href = deletePath + item\_to\_delete;

});

}

soccerDeleteDialog.sc\_deleteDialog = methods;

})(window);

Creamos una vista parcial en la carpeta **Shared** que la llamamos \_**DeleteDialog**

<div class="modal fade" id="deleteDialog" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">

<div class="modal-dialog" role="document">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">Borrar Registro</h5>

</div>

<div class="modal-body">

<p>¿Estas seguro de querer borrar el registro?</p>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-primary" data-dismiss="modal">No</button>

<button type="button" class="btn btn-danger" id="btnYesDelete">Si</button>

</div>

</div>

</div>

</div>

En **Index** modificamos:

@model IEnumerable<Vehicles.Api.Data.Entities.VehicleType>

@{

ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

<a **asp-action**="Create" class="btn btn-primary">Crear Nuevo Vehículo</a>

</p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Tipos de Vehiculo</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Description)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Description)

</td>

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/VehicleTypes/Delete/', false);

});

</script>

}

En las Vistas de **VehicleType** creamos la vista parcial **\_VehicleType**

@model Vehicles.Api.Data.Entities.VehicleType

<div class="form-group">

<label **asp-for**="Description" class="control-label"></label>

<input **asp-for**="Description" class="form-control" />

<span **asp-validation-for**="Description" class="text-danger"></span>

</div>

Y modificamos en **Create**:

@model Vehicles.Api.Data.Entities.VehicleType

@{

ViewData["Title"] = "Create";

}

<h1>Crear</h1>

<h4>Tipo de Vehículo</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Create">

<**partial** **name**="\_VehicleType" />

<div class="form-group">

<input type="submit" value="Crear" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Y modificamos en **Edit:**:

@model Vehicles.Api.Data.Entities.VehicleType

@{

ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Tipo de Vehículo</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Edit">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />|

<**partial** **name**="\_VehicleType" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

<div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

En el controlador modificamos el **Create Post**:

// POST: VehicleTypes/Create

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(VehicleType vehicleType)

{

if (ModelState.IsValid)

{

try

{

\_context.Add(vehicleType);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

catch (DbUpdateException dbUpdateException)

{

if(dbUpdateException.InnerException.Message.Contains("duplicate"))

{

ModelState.AddModelError(string.Empty, "Ya existe este tipo de vehículo.");

}

else

{

ModelState.AddModelError(string.Empty, dbUpdateException.Message);

}

}

catch (Exception exception)

{

ModelState.AddModelError(string.Empty, exception.Message);

}

}

return View(vehicleType);

}

En el controlador modificamos el **Edit Post**:

// POST: VehicleTypes/Edit/5

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, VehicleType vehicleType)

{

if (id != vehicleType.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(vehicleType);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

ModelState.AddModelError(string.Empty, "Ya existe este tipo de vehículo.");

}

else

{

ModelState.AddModelError(string.Empty, dbUpdateException.Message);

}

}

catch (Exception exception)

{

ModelState.AddModelError(string.Empty, exception.Message);

}

}

return View(vehicleType);

}

Borramos el método **VehicleTypeExists**

# 6 - Procedure

En **Entities** creamos la Clase **Procedure**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class Procedure

{

public int Id { get; set; }

[Display(Name = "Procedimiento")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public string Description { get; set; }

[Display(Name = "Precio")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public decimal Price { get; set; }

}

}

Lo agregamos en **DataContext**:

using Microsoft.EntityFrameworkCore;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Data

{

public class DataContext : DbContext

{

public DataContext(DbContextOptions<DataContext> options) : base(options)

{

}

public DbSet<Procedure> Procedures { get; set; }

public DbSet<VehicleType> VehicleTypes { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Procedure>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<VehicleType>().HasIndex(x => x.Description).IsUnique();

}

}

}

Vamos a la **Consola del Administrador de paquetes**.

Nos aseguramos que diga **Vehicles.Api** arriba y abajo

Ejecutamos los comandos:

**add-migration AddTableProcedure**

**update-database**

Creamos el Controlador **ProcedureController**

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Controllers

{

public class ProceduresController : Controller

{

private readonly DataContext \_context;

public ProceduresController(DataContext context)

{

\_context = context;

}

public async Task<IActionResult> Index()

{

return View(await \_context.Procedures.ToListAsync());

}

public IActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(Procedure procedure)

{

if (ModelState.IsValid)

{

try

{

\_context.Add(procedure);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

ModelState.AddModelError(string.Empty, "Ya existe este procedimiento.");

}

else

{

ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

ModelState.AddModelError(string.Empty, exception.Message);

}

}

return View(procedure);

}

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

Procedure procedure = await \_context.Procedures.FindAsync(id);

if (procedure == null)

{

return NotFound();

}

return View(procedure);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, Procedure procedure)

{

if (id != procedure.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(procedure);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

ModelState.AddModelError(string.Empty, "Ya existe este procedimiento.");

}

else

{

ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

ModelState.AddModelError(string.Empty, exception.Message);

}

}

return View(procedure);

}

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

Procedure procedure = await \_context.Procedures

.FirstOrDefaultAsync(m => m.Id == id);

if (procedure == null)

{

return NotFound();

}

\_context.Procedures.Remove(procedure);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

}

}

Agregamos a la Vista **Layout**:

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Procedures" **asp-action**="Index">Procedimientos</a>

</li>

Creamos las Vistas **Index**, **\_Procedure**, **Create** y **Edit**

**Index:**

@model IEnumerable<Vehicles.Api.Data.Entities.Procedure>

@{

ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

<a **asp-action**="Create" class="btn btn-primary">Nuevo</a>

</p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Tipos de Procedimientos</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Price)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Procedures/Delete/', false);

});

</script>

}

**\_ Procedure:**

@model Vehicles.Api.Data.Entities.Procedure

<div class="form-group">

<label **asp-for**="Description" class="control-label"></label>

<input **asp-for**="Description" class="form-control" />

<span **asp-validation-for**="Description" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Price" class="control-label"></label>

<input **asp-for**="Price" class="form-control" />

<span **asp-validation-for**="Price" class="text-danger"></span>

</div>

**Create:**

model Vehicles.Api.Data.Entities.Procedure

@{

ViewData["Title"] = "Create";

}

<h1>Crear</h1>

<h4>Procedimiento</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Create">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<**partial** **name**="\_Procedure" />

<div class="form-group">

<input type="submit" value="Crear" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Edit:**

@model Vehicles.Api.Data.Entities.Procedure

@{

ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Procedimiento</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Edit">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<**partial** **name**="\_Procedure" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

<div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 7 - Document Type

Hacemos lo mismo para esta Tabla

# 8 - Brand

Hacemos lo mismo para esta Tabla

# 9 - Seeder

Dentro de la carpeta **Data** creamos la Clase **SeedDb**

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Data

{

public class SeedDb

{

private readonly DataContext \_context;

public SeedDb(DataContext context)

{

\_context = context;

}

public async Task SeedAsync()

{

await \_context.Database.EnsureCreatedAsync();

await CheckVehiclesTypeAsync();

await CheckBrandsAsync();

await CheckDocumentTypesAsync();

await CheckProceduresAsync();

}

private async Task CheckProceduresAsync()

{

if (!\_context.Procedures.Any())

{

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Alineación" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lubricación de suspención delantera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lubricación de suspención trasera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Frenos delanteros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Frenos traseros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Líquido frenos delanteros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Líquido frenos traseros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Calibración de válvulas" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Alineación carburador" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Aceite motor" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Aceite caja" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Filtro de aire" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Sistema eléctrico" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Guayas" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio llanta delantera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio llanta trasera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Reparación de motor" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Kit arrastre" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Banda transmisión" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio batería" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lavado sistema de inyección" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lavada de tanque" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio de bujia" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio rodamiento delantero" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio rodamiento trasero" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Accesorios" });

await \_context.SaveChangesAsync();

}

}

private async Task CheckDocumentTypesAsync()

{

if (!\_context.DocumentTypes.Any())

{

\_context.DocumentTypes.Add(new DocumentType { Description = "DNI" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Cédula" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Libreta de Enrolamiento" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Libreta de Cívica" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Pasaporte" });

await \_context.SaveChangesAsync();

}

}

private async Task CheckBrandsAsync()

{

if (!\_context.Brands.Any())

{

\_context.Brands.Add(new Brand { Description = "Ducati" });

\_context.Brands.Add(new Brand { Description = "Harley Davidson" });

\_context.Brands.Add(new Brand { Description = "KTM" });

\_context.Brands.Add(new Brand { Description = "BMW" });

\_context.Brands.Add(new Brand { Description = "Triumph" });

\_context.Brands.Add(new Brand { Description = "Victoria" });

\_context.Brands.Add(new Brand { Description = "Honda" });

\_context.Brands.Add(new Brand { Description = "Suzuki" });

\_context.Brands.Add(new Brand { Description = "Kawasaky" });

\_context.Brands.Add(new Brand { Description = "TVS" });

\_context.Brands.Add(new Brand { Description = "Bajaj" });

\_context.Brands.Add(new Brand { Description = "AKT" });

\_context.Brands.Add(new Brand { Description = "Yamaha" });

\_context.Brands.Add(new Brand { Description = "Mazda" });

\_context.Brands.Add(new Brand { Description = "Renault" });

\_context.Brands.Add(new Brand { Description = "Chevrolet" });

\_context.Brands.Add(new Brand { Description = "Citroen" });

\_context.Brands.Add(new Brand { Description = "Fiat" });

\_context.Brands.Add(new Brand { Description = "Ford" });

\_context.Brands.Add(new Brand { Description = "Honda" });

\_context.Brands.Add(new Brand { Description = "Kia" });

\_context.Brands.Add(new Brand { Description = "Peugeot" });

\_context.Brands.Add(new Brand { Description = "Volvo" });

await \_context.SaveChangesAsync();

}

}

private async Task CheckVehiclesTypeAsync()

{

if (!\_context.VehicleTypes.Any())

{

\_context.VehicleTypes.Add(new VehicleType { Description = "Automóvil" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Moto" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Camión" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Camioneta" });

\_context.VehicleTypes.Add(new VehicleType { Description = "SUV" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Cuadriciclo" });

await \_context.SaveChangesAsync();

}

}

}

}

En **program** agregamos:

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddDbContext<DataContext>(x =>

{

x.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

});

services.AddTransient<SeedDb>();

}

Cambiamos **program** por esto:

using Microsoft.AspNetCore;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Vehicles.Api.Data;

namespace Vehicles.Api

{

public class Program

{

public static void Main(string[] args)

{

IWebHost host = CreateWebHostBuilder(args).Build();

RunSeeding(host);

host.Run();

}

private static void RunSeeding(IWebHost host)

{

IServiceScopeFactory scopeFactory = host.Services.GetService<IServiceScopeFactory>();

using (IServiceScope scope = scopeFactory.CreateScope())

{

SeedDb seeder = scope.ServiceProvider.GetService<SeedDb>();

seeder.SeedAsync().Wait();

}

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args)

{

return WebHost.CreateDefaultBuilder(args).UseStartup<Startup>();

}

}

}

# 10 – Usuarios y Roles

En el proyecto **Common** agregamos una carpeta que la llamamos **Enums**

Dentro creamos la clase **UserType**:

namespace Vehicles.Common.Enums

{

public enum UserType

{

Admin,

User

}

}

En la carpeta **wwwroot** creamos la carpeta **images**, y dentro cargamos el archivo **noimage.png** y el archivo **nouser.png**

Agregamos la referencia al Proyecto **Common** en el Proyecto Vehicles.Api

En **Data/Entities** creamos la clase **User**:

using Microsoft.AspNetCore.Identity;

using System;

using System.ComponentModel.DataAnnotations;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Data.Entities

{

public class User : IdentityUser

{

[Display(Name = "Nombres")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string FirstName { get; set; }

[Display(Name = "Apellidos")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string LastName { get; set; }

[Display(Name = "Tipo de documento")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public DocumentType DocumentType { get; set; }

[Display(Name = "Documento")]

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Document { get; set; }

[Display(Name = "Dirección")]

[MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

public string Address { get; set; }

[Display(Name = "Foto")]

public string ImageId { get; set; }

//TODO: Corregir ruta

[Display(Name = "Foto")]

public string ImageFullPath => string.IsNullOrEmpty(ImageId)

? "https://localhost:44354/images/nouser.png"

: $"https://localhost:44354{ImageId.Substring(1)}";

//: $"http://keypress.serveftp.net:88/VehiclesApi/Images/users/{ImageId}";

[Display(Name = "Tipo de usuario")]

public UserType UserType { get; set; }

[Display(Name = "Usuario")]

public string FullName => $"{FirstName} {LastName}";

}

}

En la **Entity** **DocumentType** agregamos:

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class DocumentType

{

public int Id { get; set; }

[Display(Name = "Tipo de documento")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Description { get; set; }

public ICollection<User> Users { get; set; }

}

}

En **DataContext** modificamos:

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Data

{

public class DataContext : IdentityDbContext<User>

{

public DataContext(DbContextOptions<DataContext> options) : base(options)

{

}

public DbSet<Brand> Brands { get; set; }

public DbSet<DocumentType> DocumentTypes { get; set; }

public DbSet<Procedure> Procedures { get; set; }

public DbSet<VehicleType> VehicleTypes { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Brand>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<DocumentType>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<Procedure>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<VehicleType>().HasIndex(x => x.Description).IsUnique();

}

}

}

En el proyecto **Vehicles.Api** creamos una carpeta **Helpers**

Dentro creamos la Interfaz **IUserHelper**

using Microsoft.AspNetCore.Identity;

using System.Threading.Tasks;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Helpers

{

public interface IUserHelper

{

Task<User> GetUserAsync(string email);

Task<IdentityResult> AddUserAsync(User user, string password);

Task CheckRoleAsync(string roleName);

Task AddUserToRoleAsync(User user, string roleName);

Task<bool> IsUserInRoleAsync(User user, string roleName);

}

}

Implementamos la interfaz en la Clase **UserHelper**

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Helpers

{

public class UserHelper : IUserHelper

{

private readonly UserManager<User> \_userManager;

private readonly RoleManager<IdentityRole> \_roleManager;

private readonly DataContext \_context;

private readonly SignInManager<User> \_signInManager;

public UserHelper(UserManager<User> userManager, RoleManager<IdentityRole> roleManager, DataContext context, SignInManager<User> signInManager)

{

\_userManager = userManager;

\_roleManager = roleManager;

\_context = context;

\_signInManager = signInManager;

}

public async Task<IdentityResult> AddUserAsync(User user, string password)

{

return await \_userManager.CreateAsync(user, password);

}

public async Task AddUserToRoleAsync(User user, string roleName)

{

await \_userManager.AddToRoleAsync(user, roleName);

}

public async Task CheckRoleAsync(string roleName)

{

bool roleExists = await \_roleManager.RoleExistsAsync(roleName);

if (!roleExists)

{

await \_roleManager.CreateAsync(new IdentityRole { Name = roleName });

}

}

public async Task<User> GetUserAsync(string email)

{

return await \_context.Users

.Include(x => x.DocumentType)

.FirstOrDefaultAsync(x => x.Email == email);

}

public async Task<bool> IsUserInRoleAsync(User user, string roleName)

{

return await \_userManager.IsInRoleAsync(user, roleName);

}

}

}

En **startup** agregamos:

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddIdentity<User, IdentityRole>(x =>

{

x.User.RequireUniqueEmail = true;

x.Password.RequireDigit = false;

x.Password.RequiredUniqueChars = 0;

x.Password.RequireLowercase = false;

x.Password.RequireNonAlphanumeric = false;

x.Password.RequireUppercase = false;

x.Password.RequiredLength = 6;

})

.AddEntityFrameworkStores<DataContext>();

services.AddDbContext<DataContext>(x =>

{

x.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

});

services.AddTransient<SeedDb>();

services.AddScoped<IUserHelper,UserHelper>();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseAuthentication();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

Modificamos el **SeedDb**:

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Data

{

public class SeedDb

{

private readonly DataContext \_context;

private readonly IUserHelper \_userHelper;

public SeedDb(DataContext context, IUserHelper userHelper)

{

\_context = context;

\_userHelper = userHelper;

}

public async Task SeedAsync()

{

await \_context.Database.EnsureCreatedAsync();

await CheckVehiclesTypeAsync();

await CheckBrandsAsync();

await CheckDocumentTypesAsync();

await CheckProceduresAsync();

await CheckRolesAsycn();

await CheckUserAsync("1010", "Luis", "Núñez", "luis@yopmail.com", "351 681 4963", "Espora 2052", UserType.Admin);

await CheckUserAsync("2020", "Lionel", "Messi", "messi@yopmail.com", "311 322 4620", "París", UserType.User);

await CheckUserAsync("3030", "Diego", "Maradona", "maradona@yopmail.com", "311 322 4620", "Villa Fiorito", UserType.User);

}

private async Task CheckRolesAsycn()

{

await \_userHelper.CheckRoleAsync(UserType.Admin.ToString());

await \_userHelper.CheckRoleAsync(UserType.User.ToString());

}

private async Task CheckUserAsync(string document, string firstName, string lastName, string email, string phoneNumber, string address, UserType userType)

{

User user = await \_userHelper.GetUserAsync(email);

if (user == null)

{

user = new User

{

Address = address,

Document = document,

DocumentType = \_context.DocumentTypes.FirstOrDefault(x => x.Description == "Cédula"),

Email = email,

FirstName = firstName,

LastName = lastName,

PhoneNumber = phoneNumber,

UserName = email,

UserType = userType

};

await \_userHelper.AddUserAsync(user, "123456");

await \_userHelper.AddUserToRoleAsync(user, userType.ToString());

}

}

private async Task CheckProceduresAsync()

{

if (!\_context.Procedures.Any())

{

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Alineación" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lubricación de suspención delantera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lubricación de suspención trasera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Frenos delanteros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Frenos traseros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Líquido frenos delanteros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Líquido frenos traseros" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Calibración de válvulas" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Alineación carburador" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Aceite motor" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Aceite caja" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Filtro de aire" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Sistema eléctrico" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Guayas" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio llanta delantera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio llanta trasera" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Reparación de motor" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Kit arrastre" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Banda transmisión" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio batería" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lavado sistema de inyección" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Lavada de tanque" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio de bujia" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio rodamiento delantero" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Cambio rodamiento trasero" });

\_context.Procedures.Add(new Procedure { Price = 10000, Description = "Accesorios" });

await \_context.SaveChangesAsync();

}

}

private async Task CheckDocumentTypesAsync()

{

if (!\_context.DocumentTypes.Any())

{

\_context.DocumentTypes.Add(new DocumentType { Description = "DNI" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Cédula" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Libreta de Enrolamiento" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Libreta de Cívica" });

\_context.DocumentTypes.Add(new DocumentType { Description = "Pasaporte" });

await \_context.SaveChangesAsync();

}

}

private async Task CheckBrandsAsync()

{

if (!\_context.Brands.Any())

{

\_context.Brands.Add(new Brand { Description = "Ducati" });

\_context.Brands.Add(new Brand { Description = "Harley Davidson" });

\_context.Brands.Add(new Brand { Description = "KTM" });

\_context.Brands.Add(new Brand { Description = "BMW" });

\_context.Brands.Add(new Brand { Description = "Triumph" });

\_context.Brands.Add(new Brand { Description = "Victoria" });

\_context.Brands.Add(new Brand { Description = "Suzuki" });

\_context.Brands.Add(new Brand { Description = "Kawasaky" });

\_context.Brands.Add(new Brand { Description = "TVS" });

\_context.Brands.Add(new Brand { Description = "Bajaj" });

\_context.Brands.Add(new Brand { Description = "AKT" });

\_context.Brands.Add(new Brand { Description = "Yamaha" });

\_context.Brands.Add(new Brand { Description = "Mazda" });

\_context.Brands.Add(new Brand { Description = "Renault" });

\_context.Brands.Add(new Brand { Description = "Chevrolet" });

\_context.Brands.Add(new Brand { Description = "Citroen" });

\_context.Brands.Add(new Brand { Description = "Fiat" });

\_context.Brands.Add(new Brand { Description = "Ford" });

\_context.Brands.Add(new Brand { Description = "Honda" });

\_context.Brands.Add(new Brand { Description = "Kia" });

\_context.Brands.Add(new Brand { Description = "Peugeot" });

\_context.Brands.Add(new Brand { Description = "Volvo" });

await \_context.SaveChangesAsync();

}

}

private async Task CheckVehiclesTypeAsync()

{

if (!\_context.VehicleTypes.Any())

{

\_context.VehicleTypes.Add(new VehicleType { Description = "Automóvil" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Moto" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Camión" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Camioneta" });

\_context.VehicleTypes.Add(new VehicleType { Description = "SUV" });

\_context.VehicleTypes.Add(new VehicleType { Description = "Cuadriciclo" });

await \_context.SaveChangesAsync();

}

}

}

}

Vamos a la **Consola del Administrador de paquetes**.

Nos aseguramos que diga **Vehicles.Api** arriba y abajo

Ejecutamos los comandos:

**add-migration AddUserTables**

**drop-database**

y corremos el proyecto

# 11 - Login a la Web

En la carpeta **Models** creamos la Clase **LoginViewModel**:

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class LoginViewModel

{

[Display(Name = "Email")]

[EmailAddress(ErrorMessage = "Debes introducir un email válido.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Username { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Contraseña")]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Password { get; set; }

[Display(Name = "Recordarme")]

public bool RememberMe { get; set; }

}

}

Agregamos a **IUserHelper**:

Task<SignInResult> LoginAsync(LoginViewModel model);

Task LogoutAsync();

Agregamos a **UserHelper**:

public async Task<SignInResult> LoginAsync(LoginViewModel model)

{

return await \_signInManager.PasswordSignInAsync(model.Username, model.Password, model.RememberMe, false);

}

public async Task LogoutAsync()

{

await \_signInManager.SignOutAsync();

}

Creamos el Controlador **AccountController**:

using Microsoft.AspNetCore.Mvc;

using System;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Helpers;

using Vehicles.Api.Models;

namespace Vehicles.Api.Controllers

{

public class AccountController : Controller

{

private readonly IUserHelper \_userHelper;

public AccountController(IUserHelper userHelper)

{

\_userHelper = userHelper;

}

public IActionResult Login()

{

if (User.Identity.IsAuthenticated)

{

return RedirectToAction(nameof(Index), "Home");

}

return View(new LoginViewModel());

}

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result = await \_userHelper.LoginAsync(model);

if (result.Succeeded)

{

if (Request.Query.Keys.Contains("ReturnUrl"))

{

return Redirect(Request.Query["ReturnUrl"].First());

}

return RedirectToAction("Index", "Home");

}

ModelState.AddModelError(string.Empty, "Email o contraseña incorrectos.");

}

return View(model);

}

public async Task<IActionResult> Logout()

{

await \_userHelper.LogoutAsync();

return RedirectToAction("Index", "Home");

}

}

}

Creamos la vista **Login**:

@model Vehicles.Api.Models.LoginViewModel

@{

ViewData["Title"] = "Login";

}

<h2>Inicio de Sesión</h2>

<div class="row">

<div class="col-md-6">

<form method="post">

<div **asp-validation-summary**="ModelOnly"></div>

<div class="form-group">

<label **asp-for**="Username"></label>

<input **asp-for**="Username" class="form-control" />

<span **asp-validation-for**="Username" class="text-warning"></span>

</div>

<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>

<div class="form-group">

<label **asp-for**="Password"></label>

<input **asp-for**="Password" **type**="password" class="form-control" />

<span **asp-validation-for**="Password" class="text-warning"></span>

</div>

<div class="form-group">

<div class="form-check">

<input **asp-for**="RememberMe" **type**="checkbox" class="form-check-input" />

<label **asp-for**="RememberMe" class="form-check-label"></label>

</div>

<span **asp-validation-for**="RememberMe" class="text-warning"></span>

</div>

<div class="form-group">

<input type="submit" value="Iniciar Sesión" class="btn btn-primary" />

<a **asp-action**="Register" class="btn btn-secondary">Registrar Nuevo Usuario</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Agregamos a la carpeta **wwwroot/images** el archivo **logo.png**

Modificamos la vista **Index** de **Home**:

@{

ViewData["Title"] = "Home Page";

}

<div class="text-center">

<h1 class="display-4">Vehicles</h1>

<img src="~/images/logo.png"/>

</div>

En los controladores **BrandsController**, **DocumentTypesController**, **ProceduresController** y **VehicleTypesController** agregamos:

[Authorize(Roles = "Admin")]

En la Vista **\_Layout** hacemos:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - Vehicles.Api</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container">

<a class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Vehicles</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

@if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Brands" **asp-action**="Index">Marcas</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Procedures" **asp-action**="Index">Procedimientos</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="DocumentTypes" **asp-action**="Index">Tipos de Documento</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="VehicleTypes" **asp-action**="Index">Tipos de Vehículo</a>

</li>

}

<li class="nav-item">

</li>

</ul>

<ul class="nav navbar-nav navbar-right">

@if (User.Identity.IsAuthenticated)

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Account" **asp-action**="ChangeUser">@User.Identity.Name</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Account" **asp-action**="Logout">Cerrar Sesión</a>

</li>

}

else

{

<li><a **asp-area**="" **asp-controller**="Account" **asp-action**="Login">Iniciar Sesión</a></li>

}

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2021 - Vehicles - <a **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</a>

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script **src**="~/js/site.js" **asp-append-version**="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>

# 12 – Vehicle, VehiclePhoto, History y Detail

Creamos en **Data/Entities** las Entities:

**Vehicle:**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

namespace Vehicles.Api.Data.Entities

{

public class Vehicle

{

public int Id { get; set; }

[Display(Name = "Tipo de vehículo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public VehicleType VehicleType { get; set; }

[Display(Name = "Marca")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Brand Brand { get; set; }

[Display(Name = "Modelo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[Range(1900, 3000, ErrorMessage = "Valor de modelo no válido.")]

public int Model { get; set; }

[Display(Name = "Placa")]

[RegularExpression(@"[a-zA-Z]{3}[0-9]{2}[a-zA-Z0-9]", ErrorMessage = "Formato de placa incorrecto.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[StringLength(8, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre 6 y 8 carácteres.")]

public string Plaque { get; set; }

[Display(Name = "Línea")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Line { get; set; }

[Display(Name = "Color")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Color { get; set; }

[Display(Name = "Propietario")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public User User { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

public ICollection<VehiclePhoto> VehiclePhotos { get; set; }

[Display(Name = "N° Fotos")]

public int VehiclePhotosCount => VehiclePhotos == null ? 0 : VehiclePhotos.Count;

[Display(Name = "Foto")]

public string ImageFullPath => VehiclePhotos == null || VehiclePhotos.Count == 0

? $"https://localhost:44354/images/noimage.png"

: VehiclePhotos.FirstOrDefault().ImageFullPath;

public ICollection<History> Histories { get; set; }

[Display(Name = "N° Historias")]

public int HistoriesCount => Histories == null ? 0 : Histories.Count;

}

}

**VehiclePhoto:**

using System;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class VehiclePhoto

{

public int Id { get; set; }

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Vehicle Vehicle { get; set; }

[Display(Name = "Foto")]

public string ImageId { get; set; }

[Display(Name = "Foto")]

public string ImageFullPath => ImageId == string.Empty

? $"https://localhost:44354/images/noimage.png"

: $"http://keypress.serveftp.net:88/VehiclesApi/Images/vehicles/{ImageId}";

}

}

**History:**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

namespace Vehicles.Api.Data.Entities

{

public class History

{

public int Id { get; set; }

[Display(Name = "Vehículo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Vehicle Vehicle { get; set; }

[Display(Name = "Fecha")]

[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

public DateTime Date { get; set; }

[Display(Name = "Fecha")]

[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

public DateTime DateLocal => Date.ToLocalTime();

[Display(Name = "Kilometraje")]

[DisplayFormat(DataFormatString = "{0:N0}")]

public int Mileage { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

public ICollection<Detail> Details { get; set; }

[Display(Name = "N° Detalles")]

public int DetailsCount => Details == null ? 0 : Details.Count;

[Display(Name = "Total Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal TotalLabor => Details == null ? 0 : Details.Sum(x => x.LaborPrice);

[Display(Name = "Total Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal TotalSpareParts => Details == null ? 0 : Details.Sum(x => x.SparePartsPrice);

[Display(Name = "Total")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal Total => Details == null ? 0 : Details.Sum(x => x.TotalPrice);

}

}

**Detail:**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class Detail

{

public int Id { get; set; }

[Display(Name = "Historia")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public History History { get; set; }

[Display(Name = "Procedimiento")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Procedure Procedure { get; set; }

[Display(Name = "Precio Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public decimal LaborPrice { get; set; }

[Display(Name = "Precio Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public decimal SparePartsPrice { get; set; }

[Display(Name = "Total")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal TotalPrice => LaborPrice + SparePartsPrice;

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

}

}

En **VehicleType** agregamos

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class VehicleType

{

public int Id { get; set; }

[Display(Name = "Tipo de vehículo")]

[MaxLength(50, ErrorMessage ="El campo {0} no puede tener más de {1} caracteres")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public string Description { get; set; }

public ICollection<Vehicle> Vehicles { get; set; }

}

}

En **Brands** agregamos

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Data.Entities

{

public class Brand

{

public int Id { get; set; }

[Display(Name = "Marca")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Description { get; set; }

public ICollection<Vehicle> Vehicles { get; set; }

}

}

En **User** agregamos:

using Microsoft.AspNetCore.Identity;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Data.Entities

{

public class User : IdentityUser

{

[Display(Name = "Nombres")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string FirstName { get; set; }

[Display(Name = "Apellidos")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string LastName { get; set; }

[Display(Name = "Tipo de documento")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public DocumentType DocumentType { get; set; }

[Display(Name = "Documento")]

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Document { get; set; }

[Display(Name = "Dirección")]

[MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

public string Address { get; set; }

[Display(Name = "Foto")]

public string ImageId { get; set; }

//TODO: Corregir ruta

[Display(Name = "Foto")]

public string ImageFullPath => ImageId == string.Empty

? $"https://localhost:44354/images/noimage.png"

: $"http://keypress.serveftp.net:88/VehiclesApi/Images/users/{ImageId}";

[Display(Name = "Tipo de usuario")]

public UserType UserType { get; set; }

[Display(Name = "Usuario")]

public string FullName => $"{FirstName} {LastName}";

public ICollection<Vehicle> Vehicles { get; set; }

[Display(Name = "N° Vehículos")]

public int VehiclesCount => Vehicles == null ? 0 : Vehicles.Count;

}

}

En **Procedure** agregamos:

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class Procedure

{

public int Id { get; set; }

[Display(Name = "Procedimiento")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} caracteres")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public string Description { get; set; }

[Display(Name = "Precio")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public decimal Price { get; set; }

[JsonIgnore]

public ICollection<Detail> Details { get; set; }

}

}

Dentro de la carpeta **wwwroot** creamos la carpeta **vehicles**

Agregamos en **DataContext**:

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Data

{

public class DataContext : IdentityDbContext<User>

{

public DataContext(DbContextOptions<DataContext> options) : base(options)

{

}

public DbSet<Brand> Brands { get; set; }

public DbSet<Detail> Details { get; set; }

public DbSet<DocumentType> DocumentTypes { get; set; }

public DbSet<History> Histories { get; set; }

public DbSet<Procedure> Procedures { get; set; }

public DbSet<Vehicle> Vehicles { get; set; }

public DbSet<VehiclePhoto> VehiclePhotos { get; set; }

public DbSet<VehicleType> VehicleTypes { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Entity<Brand>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<DocumentType>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<Procedure>().HasIndex(x => x.Description).IsUnique();

modelBuilder.Entity<Vehicle>().HasIndex(x => x.Plaque).IsUnique();

modelBuilder.Entity<VehicleType>().HasIndex(x => x.Description).IsUnique();

}

}

}

Vamos a la **Consola del Administrador de paquetes**.

Nos aseguramos que diga **Vehicles.Api** arriba y abajo

Ejecutamos los comandos:

**add-migration CompleteDataBase**

**drop-database**

# 13 - CombosHelper

Dentro de la Carpeta **Helpers** creamos la Interfaz **ICombosHelper**

using Microsoft.AspNetCore.Mvc.Rendering;

using System.Collections.Generic;

namespace Vehicles.Api.Helpers

{

public interface ICombosHelper

{

IEnumerable<SelectListItem> GetComboDocumentTypes();

IEnumerable<SelectListItem> GetComboProcedures();

IEnumerable<SelectListItem> GetComboVehicleTypes();

IEnumerable<SelectListItem> GetComboBrands();

}

}

Implementamos en **CombosHelper**:

using Microsoft.AspNetCore.Mvc.Rendering;

using System.Collections.Generic;

using System.Linq;

using Vehicles.Api.Data;

namespace Vehicles.Api.Helpers

{

public class CombosHelper : ICombosHelper

{

private readonly DataContext \_context;

public CombosHelper(DataContext context)

{

\_context = context;

}

public IEnumerable<SelectListItem> GetComboBrands()

{

List<SelectListItem> list = \_context.Brands.Select(x => new SelectListItem

{

Text = x.Description,

Value = $"{x.Id}"

})

.OrderBy(x => x.Text)

.ToList();

list.Insert(0, new SelectListItem

{

Text = "[Seleccione una marca...]",

Value = "0"

});

return list;

}

public IEnumerable<SelectListItem> GetComboProcedures()

{

List<SelectListItem> list = \_context.Procedures.Select(x => new SelectListItem

{

Text = x.Description,

Value = $"{x.Id}"

})

.OrderBy(x => x.Text)

.ToList();

list.Insert(0, new SelectListItem

{

Text = "[Seleccione un procedimiento...]",

Value = "0"

});

return list;

}

public IEnumerable<SelectListItem> GetComboDocumentTypes()

{

List<SelectListItem> list = \_context.DocumentTypes.Select(x => new SelectListItem

{

Text = x.Description,

Value = $"{x.Id}"

})

.OrderBy(x => x.Text)

.ToList();

list.Insert(0, new SelectListItem

{

Text = "[Seleccione un tipo de documento...]",

Value = "0"

});

return list;

}

public IEnumerable<SelectListItem> GetComboVehicleTypes()

{

List<SelectListItem> list = \_context.VehicleTypes.Select(x => new SelectListItem

{

Text = x.Description,

Value = $"{x.Id}"

})

.OrderBy(x => x.Text)

.ToList();

list.Insert(0, new SelectListItem

{

Text = "[Seleccione un tipo de vehículo...]",

Value = "0"

});

return list;

}

}

}

En **startup** ponemos:

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

services.AddIdentity<User, IdentityRole>(x =>

{

x.User.RequireUniqueEmail = true;

x.Password.RequireDigit = false;

x.Password.RequiredUniqueChars = 0;

x.Password.RequireLowercase = false;

x.Password.RequireNonAlphanumeric = false;

x.Password.RequireUppercase = false;

x.Password.RequiredLength = 6;

})

.AddEntityFrameworkStores<DataContext>();

services.AddDbContext<DataContext>(x =>

{

x.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

});

services.AddTransient<SeedDb>();

services.AddScoped<IUserHelper,UserHelper>();

services.AddScoped<ICombosHelper, CombosHelper>();

}

# 14 - UsersController

Creamos el controlador **UsersController**:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Helpers;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Controllers

{

[Authorize(Roles = "Admin")]

public class UsersController : Controller

{

private readonly DataContext \_context;

private readonly IUserHelper \_userHelper;

private readonly ICombosHelper \_combosHelper;

public UsersController(DataContext context, IUserHelper userHelper, ICombosHelper combosHelper)

{

\_context = context;

\_userHelper = userHelper;

\_combosHelper = combosHelper;

}

public async Task<IActionResult> Index()

{

return View(await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.Where(x => x.UserType == UserType.User)

.ToListAsync());

}

}

}

Creamos la vista **Index** de **Users**:

@model IEnumerable<Vehicles.Api.Data.Entities.User>

@{

ViewData["Title"] = "Index";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

<a **asp-action**="Create" class="btn btn-primary">Nuevo</a>

</p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Usuarios</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.FullName)

</th>

<th>

@Html.DisplayNameFor(model => model.DocumentType.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Document)

</th>

<th>

@Html.DisplayNameFor(model => model.Email)

</th>

<th>

@Html.DisplayNameFor(model => model.PhoneNumber)

</th>

<th>

@Html.DisplayNameFor(model => model.VehiclesCount)

</th>

<th>

@Html.DisplayNameFor(model => model.ImageFullPath)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.FullName)

</td>

<td>

@Html.DisplayFor(modelItem => item.DocumentType.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.Document)

</td>

<td>

@Html.DisplayFor(modelItem => item.Email)

</td>

<td>

@Html.DisplayFor(modelItem => item.PhoneNumber)

</td>

<td>

@Html.DisplayFor(modelItem => item.VehiclesCount)

</td>

<td>

<img src="@item.ImageFullPath" style="width: 100px; height: 100px; border-radius: 150px;" />

</td>

<td>

<a **asp-action**="Edit" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<a **asp-action**="Details" **asp-route-id**="@item.Id" class="btn btn-info">Vehiculos</a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Users/Delete/', false);

});

</script>

}

En **\_Layout** agregamos:

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Users" **asp-action**="Index">Usuarios</a>

</li>

# 15 – Imágenes

En la carpeta **Helpers** creamos la interfaz **IImageHelper**

using System.IO;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

namespace Vehicles.Api.Helpers

{

public interface IImageHelper

{

Task<string> UploadImageAsync(IFormFile imageFile, string folder);

string UploadImage(byte[] pictureArray, string folder);

Task<string> UploadImage2Async(Stream imageFile, string folder);

}

}

Y su implementación **ImageHelper**

using Microsoft.AspNetCore.Http;

using System;

using System.IO;

using System.Threading.Tasks;

namespace Vehicles.Api.Helpers

{

public class ImageHelper : IImageHelper

{

public async Task<string> UploadImageAsync(IFormFile imageFile, string folder)

{

string guid = Guid.NewGuid().ToString();

string file = $"{guid}.jpg";

string path = Path.Combine(

Directory.GetCurrentDirectory(),

$"wwwroot\\images\\{folder}",

file);

using (FileStream stream = new FileStream(path, FileMode.Create))

{

await imageFile.CopyToAsync(stream);

}

return $"~/images/{folder}/{file}";

}

public string UploadImage(byte[] pictureArray, string folder)

{

MemoryStream stream = new MemoryStream(pictureArray);

string guid = Guid.NewGuid().ToString();

string file = $"{guid}.jpg";

try

{

stream.Position = 0;

string path = Path.Combine(Directory.GetCurrentDirectory(), $"wwwroot\\images\\{folder}", file);

File.WriteAllBytes(path, stream.ToArray());

}

catch

{

return string.Empty;

}

return $"~/images/{folder}/{file}";

}

public async Task<string> UploadImage2Async(Stream imageFile, string folder)

{

string guid = Guid.NewGuid().ToString();

string file = $"{guid}.jpg";

string path = Path.Combine(

Directory.GetCurrentDirectory(),

$"wwwroot\\images\\{folder}",

file);

using (FileStream stream = new FileStream(path, FileMode.Create))

{

await imageFile.CopyToAsync(stream);

}

return $"~/images/{folder}/{file}";

}

}

}

En el Proyecto **Common** creamos una carpeta **Helpers**

Dentro creamos la Interfaz **IFilesHelper**

using System.IO;

namespace GenericApp.Common.Helpers

{

public interface IFilesHelper

{

byte[] ReadFully(Stream input);

bool UploadPhoto(MemoryStream stream, string folder, string name);

}

}

Y la implementación **FilesHelper**

using System.IO;

namespace GenericApp.Common.Helpers

{

public class FilesHelper : IFilesHelper

{

public byte[] ReadFully(Stream input)

{

using (MemoryStream ms = new MemoryStream())

{

input.CopyTo(ms);

return ms.ToArray();

}

}

public bool UploadPhoto(MemoryStream stream, string folder, string name)

{

try

{

stream.Position = 0;

var path = Path.Combine(Directory.GetCurrentDirectory(), folder, name);

File.WriteAllBytes(path, stream.ToArray());

}

catch

{

return false;

}

return true;

}

}

}

Agregamos la inyección en el **StartUp**

services.AddScoped<IImageHelper, ImageHelper>();

services.AddScoped<IFilesHelper, FilesHelper>();

# 16 – UserViewModel

Creamos dentro de la carpeta **Models** la clase **UserViewModel**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Vehicles.Api.Data.Entities;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Models

{

public class UserViewModel

{

public string Id { get; set; }

[Display(Name = "Email")]

[EmailAddress(ErrorMessage = "Debes introducir un email válido.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Email { get; set; }

[Display(Name = "Nombres")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string FirstName { get; set; }

[Display(Name = "Apellidos")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string LastName { get; set; }

[Display(Name = "Documento")]

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Document { get; set; }

[Display(Name = "Dirección")]

[MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

public string Address { get; set; }

[Display(Name = "Teléfono")]

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

public string PhoneNumber { get; set; }

[Display(Name = "Foto")]

public string ImageId { get; set; }

[Display(Name = "Tipo de usuario")]

public UserType UserType { get; set; }

[Display(Name = "Foto")]

public IFormFile ImageFile { get; set; }

[Display(Name = "Tipo de documento")]

[Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar un tipo de documento.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int DocumentTypeId { get; set; }

public IEnumerable<SelectListItem> DocumentTypes { get; set; }

[Display(Name = "Foto")]

public string ImageFullPath => string.IsNullOrEmpty(ImageId)

? "https://localhost:44354/images/nouser.png"

: $"https://localhost:44354{ImageId.Substring(1)}";

//: $"http://keypress.serveftp.net:88/VehiclesApi/Images/users/{ImageId}"; }

}

# 17 - IConverterHelper

Creamos dentro de la carpeta **Helper** la Interfaz **IConverterHelper**

using System;

using System.Threading.Tasks;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Models;

namespace Vehicles.Api.Helpers

{

public interface IConverterHelper

{

Task<User> ToUserAsync(UserViewModel model, string imageId, bool isNew);

UserViewModel ToUserViewModel(User user);

}

}

Y su implementación **ConverterHelper**

using System;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Models;

namespace Vehicles.Api.Helpers

{

public class ConverterHelper : IConverterHelper

{

private readonly DataContext \_context;

private readonly ICombosHelper \_combosHelper;

public ConverterHelper(DataContext context, ICombosHelper combosHelper)

{

\_context = context;

\_combosHelper = combosHelper;

}

public async Task<User> ToUserAsync(UserViewModel model, string imageId, bool isNew)

{

return new User

{

Address = model.Address,

Document = model.Document,

DocumentType = await \_context.DocumentTypes.FindAsync(model.DocumentTypeId),

Email = model.Email,

FirstName = model.FirstName,

Id = isNew ? Guid.NewGuid().ToString() : model.Id,

ImageId = imageId,

LastName = model.LastName,

PhoneNumber = model.PhoneNumber,

UserName = model.Email,

UserType = model.UserType,

};

}

public UserViewModel ToUserViewModel(User user)

{

return new UserViewModel

{

Address = user.Address,

Document = user.Document,

DocumentTypeId = user.DocumentType.Id,

DocumentTypes = \_combosHelper.GetComboDocumentTypes(),

Email = user.Email,

FirstName = user.FirstName,

Id = user.Id,

ImageId = user.ImageId,

LastName = user.LastName,

PhoneNumber = user.PhoneNumber,

UserType = user.UserType,

};

}

}

}

Lo inyectamos en el **StartUp**

services.AddScoped<IConverterHelper, ConverterHelper>();

# 18 – Creando un nuevo Usuario

En el **UsersController** hacemos:

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.Api.Models;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Controllers

{

[Authorize(Roles = "Admin")]

public class UsersController : Controller

{

private readonly DataContext \_context;

private readonly IUserHelper \_userHelper;

private readonly ICombosHelper \_combosHelper;

private readonly IConverterHelper \_converterHelper;

private readonly IImageHelper \_imageHelper;

public UsersController(DataContext context, IUserHelper userHelper, ICombosHelper combosHelper, IConverterHelper converterHelper, IImageHelper imageHelper)

{

\_context = context;

\_userHelper = userHelper;

\_combosHelper = combosHelper;

\_converterHelper = converterHelper;

\_imageHelper = imageHelper;

}

public async Task<IActionResult> Index()

{

return View(await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.Where(x => x.UserType == UserType.User)

.ToListAsync());

}

public IActionResult Create()

{

UserViewModel model = new UserViewModel

{

DocumentTypes = \_combosHelper.GetComboDocumentTypes()

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(UserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = string.Empty;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_converterHelper.ToUserAsync(model, imageId, true);

user.UserType = UserType.User;

await \_userHelper.AddUserAsync(user, "123456");

await \_userHelper.AddUserToRoleAsync(user, user.UserType.ToString());

return RedirectToAction(nameof(Index));

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

}

}

Creamos una vista parcial que la llamamos **\_User**:

@model Vehicles.Api.Models.UserViewModel

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label **asp-for**="FirstName" class="control-label"></label>

<input **asp-for**="FirstName" class="form-control" />

<span **asp-validation-for**="FirstName" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="LastName" class="control-label"></label>

<input **asp-for**="LastName" class="form-control">

<span **asp-validation-for**="LastName" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="DocumentTypeId" class="control-label"></label>

<select **asp-for**="DocumentTypeId" **asp-items**="Model.DocumentTypes" class="form-control"></select>

<span **asp-validation-for**="DocumentTypeId" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Document" class="control-label"></label>

<input **asp-for**="Document" class="form-control" />

<span **asp-validation-for**="Document" class="text-danger"></span>

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label **asp-for**="Address" class="control-label"></label>

<input **asp-for**="Address" class="form-control" />

<span **asp-validation-for**="Address" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="PhoneNumber" class="control-label"></label>

<input **asp-for**="PhoneNumber" class="form-control" />

<span **asp-validation-for**="PhoneNumber" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="ImageFile" class="control-label"></label>

<input **asp-for**="ImageFile" **type**="file" class="form-control" />

<span **asp-validation-for**="ImageFile" class="text-danger"></span>

</div>

</div>

</div>

Creamos la vista **Create**:

@model Vehicles.Api.Models.UserViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Crear</h2>

<h4>Usuario</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Create" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label **asp-for**="Email" class="control-label"></label>

<input **asp-for**="Email" class="form-control" />

<span **asp-validation-for**="Email" class="text-danger"></span>

</div>

<**partial** **name**="\_User" />

<div class="form-group">

<input type="submit" value="Crear" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 19 – Editar Usuario

Agregamos al **IUserHelper**

Task<User> GetUserAsync(Guid id);

Y la implementación en **UserHelper**

public async Task<User> GetUserAsync(Guid id)

{

return await \_context.Users

.Include(x => x.DocumentType)

.FirstOrDefaultAsync(x => x.Id == id.ToString());

}

Agregamos al **IUserHelper**

Task<IdentityResult> UpdateUserAsync(User user);

Y la implementación en **UserHelper**

public async Task<IdentityResult> UpdateUserAsync(User user)

{

User currentUser = await GetUserAsync(user.Email);

currentUser.LastName = user.LastName;

currentUser.FirstName = user.FirstName;

currentUser.DocumentType = user.DocumentType;

currentUser.Document = user.Document;

currentUser.Address = user.Address;

currentUser.ImageId = user.ImageId;

currentUser.PhoneNumber = user.PhoneNumber;

return await \_userManager.UpdateAsync(currentUser);

}

Agregamos al **UsersController**:

public async Task<IActionResult> Edit(string id)

{

if (string.IsNullOrEmpty(id))

{

return NotFound();

}

User user = await \_userHelper.GetUserAsync(Guid.Parse(id));

if (user == null)

{

return NotFound();

}

UserViewModel model = \_converterHelper.ToUserViewModel(user);

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(UserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = model.ImageId;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_converterHelper.ToUserAsync(model, imageId, false);

await \_userHelper.UpdateUserAsync(user);

return RedirectToAction(nameof(Index));

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

Creamos la vista **Edit**

@model Vehicles.Api.Models.UserViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Editar</h2>

<h4>Usuario</h4>

<hr />

<div class="row">

<div class="col-md-10">

<form **asp-action**="Edit" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<input **type**="hidden" **asp-for**="ImageId" />

<input **type**="hidden" **asp-for**="Email" />

<**partial** **name**="\_User" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

<div class="col-md-2">

<img src="@Model.ImageFullPath" style="width: 200px; height: 200px; border-radius: 150px;" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 20- Borrar Usuario

Agregamos al **IUserHelper**

Task<IdentityResult> DeleteUserAsync(User user);

Y la implementación en **UserHelper**

public async Task<IdentityResult> DeleteUserAsync(User user)

{

return await \_userManager.DeleteAsync(user);

}

Modificamos en **UserHelper**:

public async Task<User> GetUserAsync(string email)

{

return await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehiclePhotos)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Histories)

.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x => x.Email == email);

}

public async Task<User> GetUserAsync(Guid id)

{

return await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehiclePhotos)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Histories)

.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x => x.Id == id.ToString());

}

En el **UsersController** ponemos:

public async Task<IActionResult> Delete(string id)

{

if (string.IsNullOrEmpty(id))

{

return NotFound();

}

User user = await \_userHelper.GetUserAsync(Guid.Parse(id));

if (user == null)

{

return NotFound();

}

await \_userHelper.DeleteUserAsync(user);

return RedirectToAction(nameof(Index));

}

# 21 – Detalles de Usuario

En **UsersController** ponemos:

public async Task<IActionResult> Details(string id)

{

if (string.IsNullOrEmpty(id))

{

return NotFound();

}

User user = await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Brand)

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehicleType)

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehiclePhotos)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Histories)

.FirstOrDefaultAsync(x => x.Id == id);

if (user == null)

{

return NotFound();

}

return View(user);

}

Creamos la vista **Details**:

@model Vehicles.Api.Data.Entities.User

@{

ViewData["Title"] = "Details";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<h2>Vehículos</h2>

<div>

<h4>Usuario</h4>

<hr />

<div class="row">

<div class="col-md-8">

<dl class="row">

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.FullName)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.FullName)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.DocumentType.Description)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.DocumentType.Description)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Document)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Document)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Email)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Email)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Address)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Address)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.PhoneNumber)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.PhoneNumber)

</dd>

</dl>

</div>

<div class="col-md-4">

<img src="@Model.ImageFullPath" style="width: 200px; height: 200px; border-radius: 150px;" />

</div>

</div>

</div>

<div>

<a **asp-action**="AddVehicle" **asp-route-id**="@Model.Id" class="btn btn-primary">Adicionar Vehículo</a>

<a **asp-action**="Edit" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar Usuario</a>

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

<br />

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Vehículos</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().Plaque)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().VehicleType.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().Brand.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().Line)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().Model)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().HistoriesCount)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().VehiclePhotosCount)

</th>

<th>

@Html.DisplayNameFor(model => model.Vehicles.FirstOrDefault().ImageFullPath)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.Vehicles)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Plaque)

</td>

<td>

@Html.DisplayFor(modelItem => item.VehicleType.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.Brand.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.Line)

</td>

<td>

@Html.DisplayFor(modelItem => item.Model)

</td>

<td>

@Html.DisplayFor(modelItem => item.HistoriesCount)

</td>

<td>

@Html.DisplayFor(modelItem => item.VehiclePhotosCount)

</td>

<td>

<img src="@item.ImageFullPath" style="width:100px;height:100px;max-width: 100%; height: auto;" />

</td>

<td>

<a **asp-action**="EditVehicle" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<a **asp-action**="DetailsVehicle" **asp-route-id**="@item.Id" class="btn btn-info">Historias<i class="glyphicon glyphicon-align-justify"></i></a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Users/DeleteVehicle/', false);

});

</script>

}

# 22 – Agregar Vehículo

En la Entity **History** agregamos esta propiedad:

[JsonIgnore]

[Display(Name = "Mecánico")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public User User { get; set; }

Vamos a la **Consola del Administrador de paquetes**.

Nos aseguramos que diga **Vehicles.Api** arriba y abajo

Ejecutamos los comandos:

**drop-database**

**update-database**

Creamos en la carpeta **Models** la Clase **VehicleViewModel**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Models

{

public class VehicleViewModel

{

public int Id { get; set; }

[Display(Name = "Tipo de vehículo")]

[Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar un tipo de verhículo.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int VehicleTypeId { get; set; }

public IEnumerable<SelectListItem> VehicleTypes { get; set; }

[Display(Name = "Marca")]

[Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar una marca.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int BrandId { get; set; }

public IEnumerable<SelectListItem> Brands { get; set; }

[Display(Name = "Modelo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[Range(1900, 3000, ErrorMessage = "Valor de módelo no válido.")]

public int Model { get; set; }

[Display(Name = "Placa")]

[RegularExpression(@"[a-zA-Z]{3}[0-9]{2}[a-zA-Z0-9]", ErrorMessage = "Formato de placa incorrecto.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[StringLength(8, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener entre 6 y 8 carácteres.")]

public string Plaque { get; set; }

[Display(Name = "Línea")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Line { get; set; }

[Display(Name = "Color")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Color { get; set; }

public string UserId { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

[Display(Name = "Foto")]

public IFormFile ImageFile { get; set; }

public ICollection<VehiclePhoto> VehiclePhotos { get; set; }

}

}

En **UsersController** creamos el método **AddVehicle**

public async Task<IActionResult> AddVehicle(string id)

{

if (string.IsNullOrEmpty(id))

{

return NotFound();

}

User user = await \_context.Users

.Include(x => x.Vehicles)

.FirstOrDefaultAsync(x => x.Id == id);

if (user == null)

{

return NotFound();

}

VehicleViewModel model = new VehicleViewModel

{

Brands = \_combosHelper.GetComboBrands(),

UserId = user.Id,

VehicleTypes = \_combosHelper.GetComboVehicleTypes()

};

return View(model);

}

En **IConverterHelpers** agregamos:

Task<Vehicle> ToVehicleAsync(VehicleViewModel model, bool isNew);

VehicleViewModel ToVehicleViewModel(Vehicle vehicle);

Y la implementación en **ConverterHelpers**

public async Task<Vehicle> ToVehicleAsync(VehicleViewModel model, bool isNew)

{

return new Vehicle

{

Brand = await \_context.Brands.FindAsync(model.BrandId),

Color = model.Color,

Id = isNew ? 0 : model.Id,

Line = model.Line,

Model = model.Model,

Plaque = model.Plaque.ToUpper(),

Remarks = model.Remarks,

VehicleType = await \_context.VehicleTypes.FindAsync(model.VehicleTypeId)

};

}

public VehicleViewModel ToVehicleViewModel(Vehicle vehicle)

{

return new VehicleViewModel

{

BrandId = vehicle.Brand.Id,

Brands = \_combosHelper.GetComboBrands(),

Color = vehicle.Color,

Id = vehicle.Id,

Line = vehicle.Line,

Model = vehicle.Model,

Plaque = vehicle.Plaque.ToUpper(),

Remarks = vehicle.Remarks,

UserId = vehicle.User.Id,

VehiclePhotos = vehicle.VehiclePhotos,

VehicleTypeId = vehicle.VehicleType.Id,

VehicleTypes = \_combosHelper.GetComboVehicleTypes()

};

}

Creamos la vista parcial **\_Vehicle**

@model Vehicles.Api.Models.VehicleViewModel

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label **asp-for**="VehicleTypeId" class="control-label"></label>

<select **asp-for**="VehicleTypeId" **asp-items**="Model.VehicleTypes" class="form-control"></select>

<span **asp-validation-for**="VehicleTypeId" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="BrandId" class="control-label"></label>

<select **asp-for**="BrandId" **asp-items**="Model.Brands" class="form-control"></select>

<span **asp-validation-for**="BrandId" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Model" class="control-label"></label>

<input **asp-for**="Model" class="form-control" />

<span **asp-validation-for**="Model" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Line" class="control-label"></label>

<input **asp-for**="Line" class="form-control" />

<span **asp-validation-for**="Line" class="text-danger"></span>

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label **asp-for**="Color" class="control-label"></label>

<input **asp-for**="Color" class="form-control" />

<span **asp-validation-for**="Color" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Plaque" class="control-label"></label>

<input **asp-for**="Plaque" class="form-control">

<span **asp-validation-for**="Plaque" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Remarks" class="control-label"></label>

<textarea **asp-for**="Remarks" class="form-control"></textarea>

<span **asp-validation-for**="Remarks" class="text-danger"></span>

</div>

</div>

</div>

Creamos la vista **AddVehicle**

@model Vehicles.Api.Models.VehicleViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Crear</h2>

<h4>Vehiculo</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="AddVehicle" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="UserId" />

<**partial** **name**="\_Vehicle" />

<div class="form-group">

<label **asp-for**="ImageFile" class="control-label"></label>

<input **asp-for**="ImageFile" **type**="file" class="form-control" />

<span **asp-validation-for**="ImageFile" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Details" **asp-route-id**="@Model.UserId" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 23 - Editar Vehículo

En **UserController** hacemos:

public async Task<IActionResult> EditVehicle(int? id)

{

if (id == null)

{

return NotFound();

}

Vehicle vehicle = await \_context.Vehicles

.Include(x => x.User)

.Include(x => x.Brand)

.Include(x => x.VehicleType)

.Include(x => x.VehiclePhotos)

.FirstOrDefaultAsync(x => x.Id == id);

if (vehicle == null)

{

return NotFound();

}

VehicleViewModel model = \_converterHelper.ToVehicleViewModel(vehicle);

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> EditVehicle(int id, VehicleViewModel vehicleViewModel)

{

if (id != vehicleViewModel.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

Vehicle vehicle = await \_converterHelper.ToVehicleAsync(vehicleViewModel, false);

\_context.Vehicles.Update(vehicle);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Details), new { id = vehicleViewModel.UserId });

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

ModelState.AddModelError(string.Empty, "Ya existe un vehículo con esta placa.");

}

else

{

ModelState.AddModelError(string.Empty, dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

ModelState.AddModelError(string.Empty, exception.Message);

}

}

vehicleViewModel.Brands = \_combosHelper.GetComboBrands();

vehicleViewModel.VehicleTypes = \_combosHelper.GetComboVehicleTypes();

return View(vehicleViewModel);

}

Y hacemos la vista **EditVehicle**

@model Vehicles.Api.Models.VehicleViewModel

@{

ViewData["Title"] = "Edit";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<h1>Editar</h1>

<h4>Vehículo</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="EditVehicle">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<input **type**="hidden" **asp-for**="UserId" />

<**partial** **name**="\_Vehicle" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="AddVehicleImage" **asp-route-id**="@Model.Id" class="btn btn-info">Agregar Foto</a>

<a **asp-action**="Details" **asp-route-id**="@Model.UserId" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Fotos</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.VehiclePhotos.FirstOrDefault().ImageFullPath)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.VehiclePhotos)

{

<tr>

<td>

<img src="@item.ImageFullPath" style="width:200px;height:200px;max-width: 100%; height: auto;" />

</td>

<td>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Users/DeleteImageVehicle/', false);

});

</script>

}

# 24 – Borrar Vehículo

Agregamos a **UserController**

public async Task<IActionResult> DeleteVehicle(int? id)

{

if (id == null)

{

return NotFound();

}

Vehicle vehicle = await \_context.Vehicles

.Include(x => x.User)

.Include(x => x.VehiclePhotos)

.Include(x => x.Histories)

.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x => x.Id == id);

if (vehicle == null)

{

return NotFound();

}

\_context.Vehicles.Remove(vehicle);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Details), new { id = vehicle.User.Id });

}

# 25 – Borrar Foto de Vehículo

Agregamos a **UserController**

public async Task<IActionResult> DeleteImageVehicle(int? id)

{

if (id == null)

{

return NotFound();

}

VehiclePhoto vehiclePhoto = await \_context.VehiclePhotos

.Include(x => x.Vehicle)

.FirstOrDefaultAsync(x => x.Id == id);

if (vehiclePhoto == null)

{

return NotFound();

}

\_context.VehiclePhotos.Remove(vehiclePhoto);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(EditVehicle), new { id = vehiclePhoto.Vehicle.Id });

}

# 26 – Agregar Foto de Vehículo

En **Models** creamos la Clase **VehiclePhotoViewModel**

using Microsoft.AspNetCore.Http;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class VehiclePhotoViewModel

{

public int VehicleId { get; set; }

[Display(Name = "Foto")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public IFormFile ImageFile { get; set; }

}

}

Agregamos a **UserController**

public async Task<IActionResult> AddVehicleImage(int? id)

{

if (id == null)

{

return NotFound();

}

Vehicle vehicle = await \_context.Vehicles

.FirstOrDefaultAsync(x => x.Id == id);

if (vehicle == null)

{

return NotFound();

}

VehiclePhotoViewModel model = new()

{

VehicleId = vehicle.Id

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> AddVehicleImage(VehiclePhotoViewModel model)

{

if (ModelState.IsValid)

{

string imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "vehicles");

Vehicle vehicle = await \_context.Vehicles

.Include(x => x.VehiclePhotos)

.FirstOrDefaultAsync(x => x.Id == model.VehicleId);

if (vehicle.VehiclePhotos == null)

{

vehicle.VehiclePhotos = new List<VehiclePhoto>();

}

vehicle.VehiclePhotos.Add(new VehiclePhoto

{

ImageId = imageId

});

\_context.Vehicles.Update(vehicle);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(EditVehicle), new { id = vehicle.Id });

}

return View(model);

}

Agregamos la Vista **AddVehicleImage**

@model Vehicles.Api.Models.VehiclePhotoViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Adicionar</h2>

<h4>Foto Vehículo</h4>

<hr />

<div class="row">

<div class="col-md-6">

<form **asp-action**="AddVehicleImage" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="VehicleId" />

<div class="form-group">

<label **asp-for**="ImageFile" class="control-label"></label>

<input **asp-for**="ImageFile" **type**="file" class="form-control" />

<span **asp-validation-for**="ImageFile" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Crear" class="btn btn-primary" />

<a **asp-action**="EditVehicle" **asp-route-id**="@Model.VehicleId" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 27 – Detalles de Vehículos (Historias)

Agregamos a **UserController**

public async Task<IActionResult> DetailsVehicle(int? id)

{

if (id == null)

{

return NotFound();

}

Vehicle vehicle = await \_context.Vehicles

.Include(x => x.User)

.Include(x => x.VehicleType)

.Include(x => x.Brand)

.Include(x => x.VehiclePhotos)

.Include(x => x.Histories)

.ThenInclude(x => x.Details)

.ThenInclude(x => x.Procedure)

.Include(x => x.Histories)

.ThenInclude(x => x.User)

.FirstOrDefaultAsync(x => x.Id == id);

if (vehicle == null)

{

return NotFound();

}

return View(vehicle);

}

Creamos la Vista **DetailsVehicles**

@model Vehicles.Api.Data.Entities.Vehicle

@{

ViewData["Title"] = "Details";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<h2>Historia</h2>

<div>

<h4>Vehiculo</h4>

<hr />

<div class="row">

<div class="col-md-8">

<dl class="row">

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.VehicleType.Description)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.VehicleType.Description)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Brand.Description)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Brand.Description)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Model)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Model)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Plaque)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Plaque)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Line)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Line)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Remarks)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Remarks)

</dd>

</dl>

</div>

<div class="col-md-4">

<img src="@Model.ImageFullPath" style="width: 200px; height: 200px; max-width: 100%; height: auto;" />

</div>

</div>

</div>

<div>

<a **asp-action**="AddHistory" **asp-route-id**="@Model.Id" class="btn btn-primary">Adicionar Historia</a>

<a **asp-action**="EditVehicle" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar Vehículo</a>

<a **asp-action**="Details" **asp-route-id**="@Model.User.Id" class="btn btn-success">Regresar</a>

</div>

<br />

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Historias</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().DateLocal)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().User.FullName)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().Mileage)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().Remarks)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().DetailsCount)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().TotalLabor)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().TotalSpareParts)

</th>

<th>

@Html.DisplayNameFor(model => model.Histories.FirstOrDefault().Total)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.Histories)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.DateLocal)

</td>

<td>

@Html.DisplayFor(modelItem => item.User.FullName)

</td>

<td>

@Html.DisplayFor(modelItem => item.Mileage)

</td>

<td>

@Html.DisplayFor(modelItem => item.Remarks)

</td>

<td>

@Html.DisplayFor(modelItem => item.DetailsCount)

</td>

<td>

@Html.DisplayFor(modelItem => item.TotalLabor)

</td>

<td>

@Html.DisplayFor(modelItem => item.TotalSpareParts)

</td>

<td>

@Html.DisplayFor(modelItem => item.Total)

</td>

<td>

<a **asp-action**="EditHistory" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<a **asp-action**="DetailsHistory" **asp-route-id**="@item.Id" class="btn btn-info">Detalle<i class="glyphicon glyphicon-align-justify"></i></a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

],

"order": [[0, "desc"]]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Users/DeleteHistory/', false);

});

</script>

}

# 28 – Agregar Historia

En **Models** creamos la Clase **HistoryPhotoViewModel**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class HistoryViewModel

{

public int VehicleId { get; set; }

[Display(Name = "Kilometraje")]

[DisplayFormat(DataFormatString = "{0:N0}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int Mileage { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Remarks { get; set; }

}

}

Agregamos a **UserController**

public async Task<IActionResult> AddHistory(int? id)

{

if (id == null)

{

return NotFound();

}

Vehicle vehicle = await \_context.Vehicles.FindAsync(id);

if (vehicle == null)

{

return NotFound();

}

HistoryViewModel model = new HistoryViewModel

{

VehicleId = vehicle.Id

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> AddHistory(HistoryViewModel model)

{

if (ModelState.IsValid)

{

Vehicle vehicle = await \_context.Vehicles

.Include(x => x.Histories)

.FirstOrDefaultAsync(x => x.Id == model.VehicleId);

if (vehicle == null)

{

return NotFound();

}

User user = await \_userHelper.GetUserAsync(User.Identity.Name);

History history = new History

{

Date = DateTime.UtcNow,

Mileage = model.Mileage,

Remarks = model.Remarks,

User = user

};

if (vehicle.Histories == null)

{

vehicle.Histories = new List<History>();

}

vehicle.Histories.Add(history);

\_context.Vehicles.Update(vehicle);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(DetailsVehicle), new { id = vehicle.Id });

}

return View(model);

}

Creamos la Vista parcial **\_History**

@model Vehicles.Api.Models.HistoryViewModel

<div class="form-group">

<label **asp-for**="Mileage" class="control-label"></label>

<input **asp-for**="Mileage" class="form-control" />

<span **asp-validation-for**="Mileage" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Remarks" class="control-label"></label>

<textarea **asp-for**="Remarks" class="form-control"></textarea>

<span **asp-validation-for**="Remarks" class="text-danger"></span>

</div>

Creamos la Vista **AddHistory**

@model Vehicles.Api.Models.HistoryViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Crear</h2>

<h4>Historia</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="AddHistory" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="VehicleId" />

<**partial** **name**="\_History" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="DetailsVehicle" **asp-route-id**="@Model.VehicleId" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 29 – Editar Historia

Agregamos a **UserController**

public async Task<IActionResult> EditHistory(int? id)

{

if (id == null)

{

return NotFound();

}

History history = await \_context.Histories

.Include(x => x.Vehicle)

.FirstOrDefaultAsync(x => x.Id == id);

if (history == null)

{

return NotFound();

}

HistoryViewModel model = new HistoryViewModel

{

Mileage = history.Mileage,

Remarks = history.Remarks,

VehicleId = history.Vehicle.Id

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> EditHistory(int id, HistoryViewModel historyViewModel)

{

if (ModelState.IsValid)

{

History history = await \_context.Histories.FindAsync(id);

history.Mileage = historyViewModel.Mileage;

history.Remarks = historyViewModel.Remarks;

\_context.Histories.Update(history);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(DetailsVehicle), new { id = historyViewModel.VehicleId });

}

return View(historyViewModel);

}

Creamos la vista **EditHistory**

@model Vehicles.Api.Models.HistoryViewModel

@{

ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Historia</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="EditHistory">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="VehicleId" />

<**partial** **name**="\_History" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Index" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 30 – Borrar Historia

Agregamos a **UserController**

public async Task<IActionResult> DeleteHistory(int? id)

{

if (id == null)

{

return NotFound();

}

History history = await \_context.Histories

.Include(x => x.Details)

.Include(x => x.Vehicle)

.FirstOrDefaultAsync(x => x.Id == id);

if (history == null)

{

return NotFound();

}

\_context.Histories.Remove(history);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(DetailsVehicle), new { id = history.Vehicle.Id });

}

# 31 – Agregar Detalle a las Historias

Agregamos a **UserController**

public async Task<IActionResult> DetailsHistory(int? id)

{

if (id == null)

{

return NotFound();

}

History history = await \_context.Histories

.Include(x => x.Details)

.ThenInclude(x => x.Procedure)

.Include(x => x.Vehicle)

.ThenInclude(x => x.VehiclePhotos)

.FirstOrDefaultAsync(x => x.Id == id);

if (history == null)

{

return NotFound();

}

return View(history);

}

Creamos la Vista **DetailsHistory**

@model Vehicles.Api.Data.Entities.History

@{

ViewData["Title"] = "Details";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<h2>Historia</h2>

<div>

<h4>Historia</h4>

<hr />

<div class="row">

<div class="col-md-8">

<dl class="row">

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Vehicle.Plaque)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Vehicle.Plaque)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Date)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Date)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Mileage)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Mileage)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Remarks)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Remarks)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.TotalLabor)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.TotalLabor)

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.TotalSpareParts)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.TotalSpareParts)

</dd>

</dd>

<dt class="col-sm-3">

@Html.DisplayNameFor(model => model.Total)

</dt>

<dd class="col-sm-9">

@Html.DisplayFor(model => model.Total)

</dd>

</dl>

</div>

<div class="col-md-4">

<img src="@Model.Vehicle.ImageFullPath" style="width: 200px; height: 200px; max-width: 100%; height: auto;" />

</div>

</div>

</div>

<div>

<a **asp-action**="AddDetail" **asp-route-id**="@Model.Id" class="btn btn-primary">Adicionar Procedimiento</a>

<a **asp-action**="EditHistory" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar Historia</a>

<a **asp-action**="DetailsVehicle" **asp-route-id**="@Model.Vehicle.Id" class="btn btn-success">Regresar</a>

</div>

<br />

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Procedimientos</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Details.FirstOrDefault().Procedure.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Details.FirstOrDefault().Remarks)

</th>

<th>

@Html.DisplayNameFor(model => model.Details.FirstOrDefault().LaborPrice)

</th>

<th>

@Html.DisplayNameFor(model => model.Details.FirstOrDefault().SparePartsPrice)

</th>

<th>

@Html.DisplayNameFor(model => model.Details.FirstOrDefault().TotalPrice)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.Details)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Procedure.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.Remarks)

</td>

<td>

@Html.DisplayFor(modelItem => item.LaborPrice)

</td>

<td>

@Html.DisplayFor(modelItem => item.SparePartsPrice)

</td>

<td>

@Html.DisplayFor(modelItem => item.TotalPrice)

</td>

<td>

<a **asp-action**="EditDetail" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Users/DeleteDetail/', false);

});

</script>

}

# 32 – Adicionar Procedimiento

En **Models** creamos la Clase **DetailsViewModel**

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Models

{

public class DetailViewModel

{

public int Id { get; set; }

[Display(Name = "Precio Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public decimal LaborPrice { get; set; }

[Display(Name = "Precio Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public decimal SparePartsPrice { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

public int HistoryId { get; set; }

[Display(Name = "Procedimiento")]

[Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar un procedimiento.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int ProcedureId { get; set; }

public IEnumerable<SelectListItem> Procedures { get; set; }

}

}

En **IConverterHelper** agregamos

Task<Detail> ToDetailAsync(DetailViewModel model, bool isNew);

DetailViewModel ToDetailViewModel(Detail detail);

y su implementación **ConverterHelper**

public async Task<Detail> ToDetailAsync(DetailViewModel model, bool isNew)

{

return new Detail

{

Id = isNew ? 0 : model.Id,

History = await \_context.Histories.FindAsync(model.HistoryId),

LaborPrice = model.LaborPrice,

Procedure = await \_context.Procedures.FindAsync(model.ProcedureId),

Remarks = model.Remarks,

SparePartsPrice = model.SparePartsPrice

};

}

public DetailViewModel ToDetailViewModel(Detail detail)

{

return new DetailViewModel

{

HistoryId = detail.History.Id,

Id = detail.Id,

LaborPrice = detail.LaborPrice,

ProcedureId = detail.Procedure.Id,

Procedures = \_combosHelper.GetComboProcedures(),

Remarks = detail.Remarks,

SparePartsPrice = detail.SparePartsPrice

};

}

Agregamos a **UserController**

public async Task<IActionResult> AddDetail(int? id)

{

if (id == null)

{

return NotFound();

}

History history = await \_context.Histories.FindAsync(id);

if (history == null)

{

return NotFound();

}

DetailViewModel model = new DetailViewModel

{

HistoryId = history.Id,

Procedures = \_combosHelper.GetComboProcedures()

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> AddDetail(DetailViewModel detailViewModel)

{

if (ModelState.IsValid)

{

History history = await \_context.Histories

.Include(x => x.Details)

.FirstOrDefaultAsync(x => x.Id == detailViewModel.HistoryId);

if (history == null)

{

return NotFound();

}

if (history.Details == null)

{

history.Details = new List<Detail>();

}

Detail detail = await \_converterHelper.ToDetailAsync(detailViewModel, true);

history.Details.Add(detail);

\_context.Histories.Update(history);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(DetailsHistory), new { id = detailViewModel.HistoryId });

}

detailViewModel.Procedures = \_combosHelper.GetComboProcedures();

return View(detailViewModel);

}

Creamos la Vista parcial **\_Detail**

@model Vehicles.Api.Models.DetailViewModel

<div class="form-group">

<label **asp-for**="ProcedureId" class="control-label"></label>

<select **asp-for**="ProcedureId" **asp-items**="Model.Procedures" class="form-control"></select>

<span **asp-validation-for**="ProcedureId" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Remarks" class="control-label"></label>

<textarea **asp-for**="Remarks" class="form-control"></textarea>

<span **asp-validation-for**="Remarks" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="LaborPrice" class="control-label"></label>

<input **asp-for**="LaborPrice" class="form-control" />

<span **asp-validation-for**="LaborPrice" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="SparePartsPrice" class="control-label"></label>

<input **asp-for**="SparePartsPrice" class="form-control" />

<span **asp-validation-for**="SparePartsPrice" class="text-danger"></span>

</div>

Creamos la Vista **AddDetail**

@model Vehicles.Api.Models.DetailViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Crear</h2>

<h4>Detalle a la Historia</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="AddDetail" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="HistoryId" />

<**partial** **name**="\_Detail" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="DetailsHistory" **asp-route-id**="@Model.HistoryId" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 33 – Editar Procedimiento

Agregamos a **UserController**

public async Task<IActionResult> EditDetail(int? id)

{

if (id == null)

{

return NotFound();

}

Detail detail = await \_context.Details

.Include(x => x.History)

.Include(x => x.Procedure)

.FirstOrDefaultAsync(x => x.Id == id);

if (detail == null)

{

return NotFound();

}

DetailViewModel model = \_converterHelper.ToDetailViewModel(detail);

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> EditDetail(int id, DetailViewModel detailViewModel)

{

if (ModelState.IsValid)

{

Detail detail = await \_converterHelper.ToDetailAsync(detailViewModel, false);

\_context.Details.Update(detail);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(DetailsHistory), new { id = detailViewModel.HistoryId });

}

detailViewModel.Procedures = \_combosHelper.GetComboProcedures();

return View(detailViewModel);

}

Creamos la Vista **EditDetail**

@model Vehicles.Api.Models.DetailViewModel

@{

ViewData["Title"] = "Edit";

}

<h1>Editar</h1>

<h4>Detalle de Historia</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="EditDetail">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="HistoryId" />

<**partial** **name**="\_Detail" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="DetailsHistory" **asp-route-id**="@Model.HistoryId" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 34 – Borrar Procedimiento

Agregamos a **UserController**

public async Task<IActionResult> DeleteDetail(int? id)

{

if (id == null)

{

return NotFound();

}

Detail detail = await \_context.Details

.Include(x => x.History)

.FirstOrDefaultAsync(x => x.Id == id);

if (detail == null)

{

return NotFound();

}

\_context.Details.Remove(detail);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(DetailsHistory), new { id = detail.History.Id });

}

# 35 – Página no encontrada

En la carpeta **wwwroot/images** importamos el archivo **error404.png**

En **AccountController** ponemos:

public IActionResult NotAuthorized()

{

return View();

}

Y creamos la vista **NotAuthorized**

@{

ViewData["Title"] = "NotAuthorized";

}

<br />

<br />

<img src="~/images/error404.png" />

<h2>No estas autorizado para ejecutar esta acción!</h2>

En **StartUp** agregamos:

services.AddIdentity<User, IdentityRole>(x =>

{

x.User.RequireUniqueEmail = true;

x.Password.RequireDigit = false;

x.Password.RequiredUniqueChars = 0;

x.Password.RequireLowercase = false;

x.Password.RequireNonAlphanumeric = false;

x.Password.RequireUppercase = false;

x.Password.RequiredLength = 6;

})

.AddEntityFrameworkStores<DataContext>();

services.ConfigureApplicationCookie(options =>

{

options.LoginPath = "/Account/NotAuthorized";

options.AccessDeniedPath = "/Account/NotAuthorized";

});

services.AddDbContext<DataContext>(x =>

{

x.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"));

});

…….

app.UseStatusCodePagesWithReExecute("/error/{0}");

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseAuthentication();

app.UseRouting();

En **HomeController** agregamos

[Route("error/404")]

public IActionResult Error404()

{

return View();

}

Y creamos la vista **Error404**

@{

ViewData["Title"] = "Error404";

}

<br />

<br />

<img src="~/images/error404.png" />

<h2>Página no encontrada.</h2>

# 36 – Registrar Nuevo Usuario

Creamos en la carpeta **Models** la clase **EditUserViewModel**

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class EditUserViewModel

{

public string Id { get; set; }

[Display(Name = "Nombre")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string FirstName { get; set; }

[Display(Name = "Apellido")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string LastName { get; set; }

[Display(Name = "Tipo de documento")]

[Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar un tipo de documento.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int DocumentTypeId { get; set; }

public IEnumerable<SelectListItem> DocumentTypes { get; set; }

[Display(Name = "Documento")]

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Document { get; set; }

[Display(Name = "Dirección")]

[MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Address { get; set; }

[Display(Name = "Teléfono")]

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string PhoneNumber { get; set; }

[Display(Name = "Foto")]

public string ImageId { get; set; }

[Display(Name = "Foto")]

public string ImageFullPath => string.IsNullOrEmpty(ImageId)

? "https://localhost:44354/images/nouser.png"

: $"https://localhost:44354{ImageId.Substring(1)}";

[Display(Name = "Foto")]

public IFormFile ImageFile { get; set; }

}

}

Creamos también la Clase **AddUserViewModel**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class AddUserViewModel : EditUserViewModel

{

[Display(Name = "Email")]

[EmailAddress(ErrorMessage = "Debes introducir un email válido.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Username { get; set; }

[DataType(DataType.Password)]

[Display(Name = "Contraseña")]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Password { get; set; }

[Display(Name = "Confirmación de contraseña")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[DataType(DataType.Password)]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

[Compare("Password", ErrorMessage = "La contraseña y confirmacíón de contraseña no son iguales.")]

public string PasswordConfirm { get; set; }

}

}

En **IUserHelper** agregamos:

Task<User> AddUserAsync(AddUserViewModel model, string imageId, UserType userType);

Y su implementación en **UserHelper**:

public async Task<User> AddUserAsync(AddUserViewModel model, string imageId, UserType userType)

{

User user = new User

{

Address = model.Address,

Document = model.Document,

Email = model.Username,

FirstName = model.FirstName,

LastName = model.LastName,

ImageId = imageId,

PhoneNumber = model.PhoneNumber,

DocumentType = await \_context.DocumentTypes.FindAsync(model.DocumentTypeId),

UserName = model.Username,

UserType = userType

};

IdentityResult result = await \_userManager.CreateAsync(user, model.Password);

if (result != IdentityResult.Success)

{

return null;

}

User newUser = await GetUserAsync(model.Username);

await AddUserToRoleAsync(newUser, user.UserType.ToString());

return newUser;

}

En **AccountController** hacemos:

public IActionResult Register()

{

AddUserViewModel model = new AddUserViewModel

{

DocumentTypes = \_combosHelper.GetComboDocumentTypes()

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(AddUserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = string.Empty;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_userHelper.AddUserAsync(model, imageId, UserType.User);

if (user == null)

{

ModelState.AddModelError(string.Empty, "Este correo ya está siendo usado por otro usuario.");

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

LoginViewModel loginViewModel = new LoginViewModel

{

Password = model.Password,

RememberMe = false,

Username = model.Username

};

var result2 = await \_userHelper.LoginAsync(loginViewModel);

if(result2.Succeeded)

{

return RedirectToAction("Index", "Home");

}

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

Creamos la vista parcial **\_User**

@model Vehicles.Api.Models.EditUserViewModel

<div class="row">

<div class="col-md-6">

<div class="form-group">

<label **asp-for**="FirstName" class="control-label"></label>

<input **asp-for**="FirstName" class="form-control" />

<span **asp-validation-for**="FirstName" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="LastName" class="control-label"></label>

<input **asp-for**="LastName" class="form-control" />

<span **asp-validation-for**="LastName" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="DocumentTypeId" class="control-label"></label>

<select **asp-for**="DocumentTypeId" **asp-items**="Model.DocumentTypes" class="form-control"></select>

<span **asp-validation-for**="DocumentTypeId" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="Document" class="control-label"></label>

<input **asp-for**="Document" class="form-control" />

<span **asp-validation-for**="Document" class="text-danger"></span>

</div>

</div>

<div class="col-md-6">

<div class="form-group">

<label **asp-for**="Address" class="control-label"></label>

<input **asp-for**="Address" class="form-control" />

<span **asp-validation-for**="Address" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="PhoneNumber" class="control-label"></label>

<input **asp-for**="PhoneNumber" class="form-control" />

<span **asp-validation-for**="PhoneNumber" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="ImageFile" class="control-label"></label>

<input **asp-for**="ImageFile" class="form-control" **type**="file" />

<span **asp-validation-for**="ImageFile" class="text-danger"></span>

</div>

</div>

</div>

Y creamos la vista **Register**

@model Vehicles.Api.Models.AddUserViewModel

@{

ViewData["Title"] = "Register";

}

<h2>Registrar</h2>

<h4>Usuario</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Register" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label **asp-for**="Username" class="control-label"></label>

<input **asp-for**="Username" class="form-control" />

<span **asp-validation-for**="Username" class="text-danger"></span>

</div>

<**partial** **name**="\_User" />

<div class="form-group">

<label **asp-for**="Password" class="control-label"></label>

<input **asp-for**="Password" class="form-control" />

<span **asp-validation-for**="Password" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="PasswordConfirm" class="control-label"></label>

<input **asp-for**="PasswordConfirm" class="form-control" />

<span **asp-validation-for**="PasswordConfirm" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Registrar" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div class="text-success">

<p>

@ViewBag.Message

</p>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 37 – Editar Usuario y cambiar Password

Creamos en la carpeta **Models** la clase **ChangePasswordViewModel**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class ChangePasswordViewModel

{

[Display(Name = "Contraseña actual")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[DataType(DataType.Password)]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

public string OldPassword { get; set; }

[Display(Name = "Nueva contraseña")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[DataType(DataType.Password)]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

public string NewPassword { get; set; }

[Display(Name = "Confirmación de contraseña")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[DataType(DataType.Password)]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

[Compare("NewPassword", ErrorMessage = "La nueva contraseña y la confirmación no son iguales.")]

public string Confirm { get; set; }

}

}

En **IUserHelper** agregamos:

Task<IdentityResult> ChangePasswordAsync(User user, string oldPassword, string newPassword);

Y su implementación en **UserHelper**:

public async Task<IdentityResult> ChangePasswordAsync(User user, string oldPassword, string newPassword)

{

return await \_userManager.ChangePasswordAsync(user, oldPassword, newPassword);

}

En el **AccountController** hacemos:

public async Task<IActionResult> ChangeUser()

{

User user = await \_userHelper.GetUserAsync(User.Identity.Name);

if (user == null)

{

return NotFound();

}

EditUserViewModel model = new()

{

Address = user.Address,

FirstName = user.FirstName,

LastName = user.LastName,

PhoneNumber = user.PhoneNumber,

ImageId = user.ImageId,

Id = user.Id,

Document = user.Document,

DocumentTypeId = user.DocumentType.Id,

DocumentTypes = \_combosHelper.GetComboDocumentTypes(),

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> ChangeUser(EditUserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = model.ImageId;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_userHelper.GetUserAsync(User.Identity.Name);

user.FirstName = model.FirstName;

user.LastName = model.LastName;

user.Address = model.Address;

user.PhoneNumber = model.PhoneNumber;

user.ImageId = imageId;

user.DocumentType = await \_context.DocumentTypes.FindAsync(model.DocumentTypeId);

user.Document = model.Document;

await \_userHelper.UpdateUserAsync(user);

return RedirectToAction("Index", "Home");

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

public IActionResult ChangePassword()

{

return View();

}

[HttpPost]

public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userHelper.GetUserAsync(User.Identity.Name);

if (user != null)

{

IdentityResult result = await \_userHelper.ChangePasswordAsync(user, model.OldPassword, model.NewPassword);

if (result.Succeeded)

{

return RedirectToAction(nameof(ChangeUser));

}

else

{

ModelState.AddModelError(string.Empty, result.Errors.FirstOrDefault().Description);

}

}

else

{

ModelState.AddModelError(string.Empty, "Usuario no encontrado.");

}

}

return View(model);

}

Creamos la vista **ChangeUser**

@model Vehicles.Api.Models.EditUserViewModel

@{

ViewData["Title"] = "Edit";

}

<h2>Editar</h2>

<h4>Usuario</h4>

<hr />

<div class="row">

<div class="col-md-8">

<form **asp-action**="ChangeUser" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<input **type**="hidden" **asp-for**="ImageId" />

<**partial** **name**="\_User" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="ChangePassword" class="btn btn-warning">Cambiar Contraseña</a>

</div>

</form>

</div>

<div class="col-md-4">

<img src="@Model.ImageFullPath" style="width:250px;height:250px;border-radius:50%" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Creamos la vista **ChangePassword**

@model Vehicles.Api.Models.ChangePasswordViewModel

@{

ViewData["Title"] = "Register";

}

<h2>Cambiar Contraseña</h2>

<div class="row">

<div class="col-md-6">

<form method="post">

<div **asp-validation-summary**="ModelOnly"></div>

<div class="form-group">

<label **asp-for**="OldPassword"></label>

<input **asp-for**="OldPassword" **type**="password" class="form-control" />

<span **asp-validation-for**="OldPassword" class="text-warning"></span>

</div>

<div class="form-group">

<label **asp-for**="NewPassword"></label>

<input **asp-for**="NewPassword" **type**="password" class="form-control" />

<span **asp-validation-for**="NewPassword" class="text-warning"></span>

</div>

<div class="form-group">

<label **asp-for**="Confirm"></label>

<input **asp-for**="Confirm" **type**="password" class="form-control" />

<span **asp-validation-for**="Confirm" class="text-warning"></span>

</div>

<div class="form-group">

<input type="submit" value="Cambiar Contraseña" class="btn btn-primary" />

<a **asp-action**="ChangeUser" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 38 – Validar Email al registrarse

En **StartUp** hacemos

services.AddIdentity<User, IdentityRole>(x =>

{

x.Tokens.AuthenticatorTokenProvider = TokenOptions.DefaultAuthenticatorProvider;

x.SignIn.RequireConfirmedEmail = true;

x.User.RequireUniqueEmail = true;

x.Password.RequireDigit = false;

x.Password.RequiredUniqueChars = 0;

x.Password.RequireLowercase = false;

x.Password.RequireNonAlphanumeric = false;

x.Password.RequireUppercase = false;

x.Password.RequiredLength = 6;

})

.AddDefaultTokenProviders()

.AddEntityFrameworkStores<DataContext>();

services.ConfigureApplicationCookie(options =>

{

options.LoginPath = "/Account/NotAuthorized";

options.AccessDeniedPath = "/Account/NotAuthorized";

});

En **appsettings.json agregamos**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": { "DefaultConnection": "Server=keypress.serveftp.net;Database=LuisVehicles;User Id=sa;password=sentey14$;Trusted\_Connection=False;MultipleActiveResultSets=true" },

"Mail": {

"From": "luis.solflix@gmail.com",

"Smtp": "smtp.gmail.com",

"Port": 587,

"Password": "Solflix2306"

}

}

En la cuenta de **Gmail** entramos a la web https://myaccount.google.com/lesssecureapps

y tenemos que habilitar



Y entramos a la web https://accounts.google.com/DisplayUnlockCaptcha

y habilitamos lo siguiente:



En el Proyecto **Common** creamos la carpeta **Models** y dentro creamos la Clase **Response**

namespace Vehicles.Common.Models

{

public class Response

{

public bool IsSuccess { get; set; }

public string Message { get; set; }

public object Result { get; set; }

}

}

En el Proyecto **Vehicles.Api** dentro de la carpeta **Helpers** la Interfaz **IMailHelper**

using Vehicles.Common.Models;

namespace Vehicles.Api.Helpers

{

public interface IMailHelper

{

Response SendMail(string to, string subject, string body);

}

}

Agregamos al Proyecto **Vehicles.Api** el Nugget **MailKit versión 2.15.0**

Y creamos la implementación **MailHelper**

using MailKit.Net.Smtp;

using Microsoft.Extensions.Configuration;

using MimeKit;

using System;

using Vehicles.Common.Models;

namespace Vehicles.Api.Helpers

{

public class MailHelper : IMailHelper

{

private readonly IConfiguration \_configuration;

public MailHelper(IConfiguration configuration)

{

\_configuration = configuration;

}

public Response SendMail(string to, string subject, string body)

{

try

{

string from = \_configuration["Mail:From"];

string smtp = \_configuration["Mail:Smtp"];

string port = \_configuration["Mail:Port"];

string password = \_configuration["Mail:Password"];

MimeMessage message = new MimeMessage();

message.From.Add(new MailboxAddress(from));

message.To.Add(new MailboxAddress(to));

message.Subject = subject;

BodyBuilder bodyBuilder = new BodyBuilder

{

HtmlBody = body

};

message.Body = bodyBuilder.ToMessageBody();

using (SmtpClient client = new SmtpClient())

{

client.Connect(smtp, int.Parse(port), false);

client.Authenticate(from, password);

client.Send(message);

client.Disconnect(true);

}

return new Response { IsSuccess = true };

}

catch (Exception ex)

{

return new Response

{

IsSuccess = false,

Message = ex.Message,

Result = ex

};

}

}

}

}

Y la ponemos en **StartUp**

services.AddTransient<SeedDb>();

services.AddScoped<IUserHelper,UserHelper>();

services.AddScoped<ICombosHelper, CombosHelper>();

services.AddScoped<IImageHelper, ImageHelper>();

services.AddScoped<IFilesHelper, FilesHelper>();

services.AddScoped<IConverterHelper, ConverterHelper>();

services.AddScoped<IMailHelper, MailHelper>();

Agregamos a **IUserHelper**

Task<string> GenerateEmailConfirmationTokenAsync(User user);

Task<IdentityResult> ConfirmEmailAsync(User user, string token);

Y la implementación en **UserHelper**

public async Task<IdentityResult> ConfirmEmailAsync(User user, string token)

{

return await \_userManager.ConfirmEmailAsync(user, token);

}

public async Task<string> GenerateEmailConfirmationTokenAsync(User user)

{

return await \_userManager.GenerateEmailConfirmationTokenAsync(user);

}

En AccountController hacemos:

public class AccountController : Controller

{

private readonly IUserHelper \_userHelper;

private readonly DataContext \_context;

private readonly ICombosHelper \_combosHelper;

private readonly IImageHelper \_imageHelper;

private readonly IMailHelper \_mailHelper;

public AccountController(IUserHelper userHelper, DataContext context, ICombosHelper combosHelper, IImageHelper blobHelper, IMailHelper mailHelper)

{

\_userHelper = userHelper;

\_context = context;

\_combosHelper = combosHelper;

\_imageHelper = blobHelper;

\_mailHelper = mailHelper;

}

……………

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Register(AddUserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = string.Empty;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_userHelper.AddUserAsync(model, imageId, UserType.User);

if (user == null)

{

ModelState.AddModelError(string.Empty, "Este correo ya está siendo usado por otro usuario.");

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

//LoginViewModel loginViewModel = new LoginViewModel

//{

// Password = model.Password,

// RememberMe = false,

// Username = model.Username

//};

//var result2 = await \_userHelper.LoginAsync(loginViewModel);

//if(result2.Succeeded)

//{

// return RedirectToAction("Index", "Home");

//}

string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

string tokenLink = Url.Action("ConfirmEmail", "Account", new

{

userid = user.Id,

token = myToken

}, protocol: HttpContext.Request.Scheme);

Response response = \_mailHelper.SendMail(model.Username, "Vehicles - Confirmación de cuenta", $"<h1>Vehicles - Confirmación de cuenta</h1>" +

$"Para habilitar el usuario, " +

$"por favor hacer clic en el siguiente enlace: </br></br><a href = \"{tokenLink}\">Confirmar Email</a>");

if (response.IsSuccess)

{

ViewBag.Message = "Las instrucciones para habilitar su cuenta han sido enviadas al correo.";

return View(model);

}

ModelState.AddModelError(string.Empty, response.Message);

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

Agregamos a **AccountController**:

public async Task<IActionResult> ConfirmEmail(string userId, string token)

{

if (string.IsNullOrEmpty(userId) || string.IsNullOrEmpty(token))

{

return NotFound();

}

User user = await \_userHelper.GetUserAsync(new Guid(userId));

if (user == null)

{

return NotFound();

}

IdentityResult result = await \_userHelper.ConfirmEmailAsync(user, token);

if (!result.Succeeded)

{

return NotFound();

}

return View();

}

Y creamos la vista **ConfirmEmail**

@{

ViewData["Title"] = "Confirmación de Email";

}

<h2>@ViewData["Title"]</h2>

<div>

<p><strong>Gracias por confirmar su cuenta. Ya puedes ingresar al sistema.</strong></p>

<p>Si no has asignado una contraseña, tu contraseña actual es: 123456, recuerde cambiarla por una más segura y de fácil recordación para usted.</p>

</div>

En **SeedDb** agregamos:

private async Task CheckUserAsync(string document, string firstName, string lastName, string email, string phoneNumber, string address, UserType userType)

{

User user = await \_userHelper.GetUserAsync(email);

if (user == null)

{

user = new User

{

Address = address,

Document = document,

DocumentType = \_context.DocumentTypes.FirstOrDefault(x => x.Description == "Cédula"),

Email = email,

FirstName = firstName,

LastName = lastName,

PhoneNumber = phoneNumber,

UserName = email,

UserType = userType

};

await \_userHelper.AddUserAsync(user, "123456");

await \_userHelper.AddUserToRoleAsync(user, userType.ToString());

string token = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

await \_userHelper.ConfirmEmailAsync(user, token);

}

}

En **UserController** hacemos:

public class UsersController : Controller

{

private readonly DataContext \_context;

private readonly IUserHelper \_userHelper;

private readonly ICombosHelper \_combosHelper;

private readonly IConverterHelper \_converterHelper;

private readonly IImageHelper \_imageHelper;

private readonly IMailHelper \_mailHelper;

public UsersController(DataContext context, IUserHelper userHelper, ICombosHelper combosHelper, IConverterHelper converterHelper, IImageHelper imageHelper,IMailHelper mailHelper)

{

\_context = context;

\_userHelper = userHelper;

\_combosHelper = combosHelper;

\_converterHelper = converterHelper;

\_imageHelper = imageHelper;

\_mailHelper = mailHelper;

}

……..

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create(UserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = string.Empty;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_converterHelper.ToUserAsync(model, imageId, true);

user.UserType = UserType.User;

await \_userHelper.AddUserAsync(user, "123456");

await \_userHelper.AddUserToRoleAsync(user, user.UserType.ToString());

string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

string tokenLink = Url.Action("ConfirmEmail", "Account", new

{

userid = user.Id,

token = myToken

}, protocol: HttpContext.Request.Scheme);

Response response = \_mailHelper.SendMail(model.Email, "Vehicles - Confirmación de cuenta", $"<h1>Vehicles - Confirmación de cuenta</h1>" +

$"Para habilitar el usuario, " +

$"por favor hacer clic en el siguiente enlace: </br></br><a href = \"{tokenLink}\">Confirmar Email</a>");

return RedirectToAction(nameof(Index));

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

# 39 – Recuperar Contraseña

Modificamos la vista **Login**

<div class="form-group">

<input type="submit" value="Iniciar Sesión" class="btn btn-primary" />

<a **asp-action**="Register" class="btn btn-secondary">Registrar Nuevo Usuario</a>

<hr />

<a **asp-action**="RecoverPassword" class="btn btn-link">¿Olvidaste tu contraseña?</a>

</div>

En **IUserHelper** agregamos:

Task<string> GeneratePasswordResetTokenAsync(User user);

Task<IdentityResult> ResetPasswordAsync(User user, string token, string password);

Y la implementación en **UserHelper**:

public async Task<string> GeneratePasswordResetTokenAsync(User user)

{

return await \_userManager.GeneratePasswordResetTokenAsync(user);

}

public async Task<IdentityResult> ResetPasswordAsync(User user, string token, string password)

{

return await \_userManager.ResetPasswordAsync(user, token, password);

}

Creamos en la carpeta **Models** la clase **RecoverPasswordViewModel**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class RecoverPasswordViewModel

{

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[EmailAddress(ErrorMessage = "Debes introducir un email válido.")]

public string Email { get; set; }

}

}

Creamos en la carpeta **Models** la clase **ResetPasswordViewModel**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.Api.Models

{

public class ResetPasswordViewModel

{

[Display(Name = "Email")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[EmailAddress(ErrorMessage = "Debes introducir un email válido.")]

public string UserName { get; set; }

[Display(Name = "Nueva contraseña")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Confirmación de contraseña")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

[DataType(DataType.Password)]

[Compare("Password", ErrorMessage = "La nueva contraseña y la confirmación no son iguales.")]

public string ConfirmPassword { get; set; }

[Required]

public string Token { get; set; }

}

}

En **AccountController** hacemos:

public IActionResult RecoverPassword()

{

return View();

}

[HttpPost]

public async Task<IActionResult> RecoverPassword(RecoverPasswordViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userHelper.GetUserAsync(model.Email);

if (user == null)

{

ModelState.AddModelError(string.Empty, "El correo ingresado no corresponde a ningún usuario.");

return View(model);

}

string myToken = await \_userHelper.GeneratePasswordResetTokenAsync(user);

string link = Url.Action(

"ResetPassword",

"Account",

new { token = myToken }, protocol: HttpContext.Request.Scheme);

\_mailHelper.SendMail(model.Email, "Vehicles - Reseteo de contraseña", $"<h1>Vehicles - Reseteo de contraseña</h1>" +

$"Para establecer una nueva contraseña haga clic en el siguiente enlace:</br></br>" +

$"<a href = \"{link}\">Cambio de Contraseña</a>");

ViewBag.Message = "Las instrucciones para el cambio de contraseña han sido enviadas a su email.";

return View();

}

return View(model);

}

public IActionResult ResetPassword(string token)

{

return View();

}

[HttpPost]

public async Task<IActionResult> ResetPassword(ResetPasswordViewModel model)

{

User user = await \_userHelper.GetUserAsync(model.UserName);

if (user != null)

{

IdentityResult result = await \_userHelper.ResetPasswordAsync(user, model.Token, model.Password);

if (result.Succeeded)

{

ViewBag.Message = "Contaseña cambiada.";

return View();

}

ViewBag.Message = "Error cambiando la contraseña.";

return View(model);

}

ViewBag.Message = "Usuario no encontrado.";

return View(model);

}

Y creamos la vista **RecoverPassword**

@model Vehicles.Api.Models.RecoverPasswordViewModel

@{

ViewData["Title"] = "Recover Password";

}

<h2>Recuperar Contraseña</h2>

<div class="row">

<div class="col-md-6">

<form method="post">

<div **asp-validation-summary**="ModelOnly"></div>

<div class="form-group">

<label **asp-for**="Email"></label>

<input **asp-for**="Email" class="form-control" />

<span **asp-validation-for**="Email" class="text-warning"></span>

</div>

<div class="form-group">

<input type="submit" value="Recuperar Contraseña" class="btn btn-primary" />

<a **asp-action**="Login" class="btn btn-success">Regresar</a>

</div>

</form>

<div class="text-success">

<p>

@ViewBag.Message

</p>

</div>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Y creamos la vista **ResetPassword**

@model Vehicles.Api.Models.ResetPasswordViewModel

@{

ViewData["Title"] = "Reset Password";

}

<h1>Cambiar Contraseña</h1>

<div class="row">

<div class="col-md-6">

<form method="post">

<div **asp-validation-summary**="All"></div>

<input **type**="hidden" **asp-for**="Token" />

<div class="form-group">

<label **asp-for**="UserName"></label>

<input **asp-for**="UserName" class="form-control" />

<span **asp-validation-for**="UserName" class="text-warning"></span>

</div>

<div class="form-group">

<label **asp-for**="Password"></label>

<input **asp-for**="Password" **type**="password" class="form-control" />

<span **asp-validation-for**="Password" class="text-warning"></span>

</div>

<div class="form-group">

<label **asp-for**="ConfirmPassword"></label>

<input **asp-for**="ConfirmPassword" **type**="password" class="form-control" />

<span **asp-validation-for**="ConfirmPassword" class="text-warning"></span>

</div>

<div class="form-group">

<input type="submit" value="Cambiar Contraseña" class="btn btn-primary" />

</div>

</form>

<div class="text-success">

<p>

@ViewBag.Message

</p>

</div>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# 40 – API Controllers

Dentro de **Controllers** creamos una Carpeta **API**

Creamos un controlador de **Procedures**



using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

public class ProceduresController : ControllerBase

{

private readonly DataContext \_context;

public ProceduresController(DataContext context)

{

\_context = context;

}

// GET: api/Procedures

[HttpGet]

public async Task<ActionResult<IEnumerable<Procedure>>> GetProcedures()

{

return await \_context.Procedures.ToListAsync();

}

// GET: api/Procedures/5

[HttpGet("{id}")]

public async Task<ActionResult<Procedure>> GetProcedure(int id)

{

Procedure procedure = await \_context.Procedures.FindAsync(id);

if (procedure == null)

{

return NotFound();

}

return procedure;

}

// PUT: api/Procedures/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutProcedure(int id, Procedure procedure)

{

if (id != procedure.Id)

{

return BadRequest();

}

\_context.Entry(procedure).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest ("Ya existe este procedimiento.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

return NoContent();

}

// POST: api/Procedures

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Procedure>> PostProcedure(Procedure procedure)

{

\_context.Procedures.Add(procedure);

try

{

await \_context.SaveChangesAsync();

return CreatedAtAction("GetProcedure", new { id = procedure.Id }, procedure);

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe este procedimiento.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// DELETE: api/Procedures/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteProcedure(int id)

{

Procedure procedure = await \_context.Procedures.FindAsync(id);

if (procedure == null)

{

return NotFound();

}

\_context.Procedures.Remove(procedure);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool ProcedureExists(int id)

{

return \_context.Procedures.Any(e => e.Id == id);

}

}

}

Corremos la aplicación y vamos a Postman

En Postman ponemos:

Para traer el listado de Procedures:

<https://localhost:44354/api/procedures>

Para traer un Procedure:

https://localhost:44354/api/procedures/1

Para crear un Procedure:



Para modificar un Procedure:



Para borrar un Procedure:



# 41 – Seguridad en la API

Agregamos en **ProceduresController**

namespace Vehicles.Api.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

public class ProceduresController : ControllerBase

{

private readonly DataContext \_context;

public ProceduresController(DataContext context)

{

\_context = context;

}

En **appsettings** agregamos:

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*",

"ConnectionStrings": { "DefaultConnection": "Server=keypress.serveftp.net;Database=LuisVehicles;User Id=sa;password=sentey14$;Trusted\_Connection=False;MultipleActiveResultSets=true" },

"Mail": {

"From": "luis.solflix@gmail.com",

"Smtp": "smtp.gmail.com",

"Port": 587,

"Password": "Solflix2306"

},

"Tokens": {

"Key": "dalsdjlaj7997&%$#)=?9hasdlakdjkuoiqwJKJK)/()/LJLJLnm,mskldfjssljlsdKSÑDFFSsdfsdfnvDFKLÑWIOEjklnlsfdklsjfiuisouJLKHLLA",

"Issuer": "localhost",

"Audience": "users"

}

}

En **IUserHelper** agregamos:

Task<SignInResult> ValidatePasswordAsync(User user, string password);

Y la implementación en **UserHelper**:

public async Task<SignInResult> ValidatePasswordAsync(User user, string password)

{

return await \_signInManager.CheckPasswordSignInAsync(user, password, false);

}

Dentro de **Controllers/API** creamos el **AccountController**

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Tokens;

using System;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.Api.Models;

namespace Vehicles.API.Controllers.API

{

[ApiController]

[Route("api/[controller]")]

public class AccountController : ControllerBase

{

private readonly IUserHelper \_userHelper;

private readonly IConfiguration \_configuration;

public AccountController(IUserHelper userHelper, IConfiguration configuration)

{

\_userHelper = userHelper;

\_configuration = configuration;

}

[HttpPost]

[Route("CreateToken")]

public async Task<IActionResult> CreateToken([FromBody] LoginViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userHelper.GetUserAsync(model.Username);

if (user != null)

{

var result = await \_userHelper.ValidatePasswordAsync(user, model.Password);

if (result.Succeeded)

{

Claim[] claims = new[]

{

new Claim(JwtRegisteredClaimNames.Sub, user.Email),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())

};

SymmetricSecurityKey key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Tokens:Key"]));

SigningCredentials credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

JwtSecurityToken token = new JwtSecurityToken(

\_configuration["Tokens:Issuer"],

\_configuration["Tokens:Audience"],

claims,

expires: DateTime.UtcNow.AddDays(99),

signingCredentials: credentials);

var results = new

{

token = new JwtSecurityTokenHandler().WriteToken(token),

expiration = token.ValidTo,

user

};

return Created(string.Empty, results);

}

}

}

return BadRequest();

}

}

}

En **StartUp** hacemos:

services.AddIdentity<User, IdentityRole>(x =>

{

x.Tokens.AuthenticatorTokenProvider = TokenOptions.DefaultAuthenticatorProvider;

x.SignIn.RequireConfirmedEmail = true;

x.User.RequireUniqueEmail = true;

x.Password.RequireDigit = false;

x.Password.RequiredUniqueChars = 0;

x.Password.RequireLowercase = false;

x.Password.RequireNonAlphanumeric = false;

x.Password.RequireUppercase = false;

x.Password.RequiredLength = 6;

})

.AddDefaultTokenProviders()

.AddEntityFrameworkStores<DataContext>();

services.AddAuthentication()

.AddCookie()

.AddJwtBearer(cfg =>

{

cfg.TokenValidationParameters = new TokenValidationParameters

{

ValidIssuer = Configuration["Tokens:Issuer"],

ValidAudience = Configuration["Tokens:Audience"],

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(Configuration["Tokens:Key"]))

};

});

services.ConfigureApplicationCookie(options =>

{

options.LoginPath = "/Account/NotAuthorized";

options.AccessDeniedPath = "/Account/NotAuthorized";

});

En la Entity **DocumentType** hacemos:

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class DocumentType

{

public int Id { get; set; }

[Display(Name = "Tipo de documento")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Description { get; set; }

[JsonIgnore]

public ICollection<User> Users { get; set; }

}

}

En **Vehicle** lo ponemos a **User**:

[Display(Name = "Propietario")]

[JsonIgnore]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public User User { get; set; }

En **Detail** lo ponemos en **History**

[JsonIgnore]

[Display(Name = "Historia")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public History History { get; set; }

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**No me anda los del JsonIgnore. Lo solucioné así:**

Creo en **Models** la clase **UserResponse**

using Vehicles.Api.Data.Entities;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Models

{

public class UserResponse

{

public string FirstName { get; set; }

public string LastName { get; set; }

public DocumentType DocumentType { get; set; }

public string Document { get; set; }

public string Address { get; set; }

public string ImageId { get; set; }

public string PhoneNumber { get; set; }

public string Email { get; set; }

public string Id { get; set; }

public int DocumentTypeId { get; set; }

//TODO: Corregir ruta

public string ImageFullPath => string.IsNullOrEmpty(ImageId)

? "https://localhost:44354/images/nouser.png"

: $"https://localhost:44354{ImageId.Substring(1)}";

//: $"http://keypress.serveftp.net:88/VehiclesApi/Images/users/{ImageId}";

public UserType UserType { get; set; }

public string FullName => $"{FirstName} {LastName}";

}

}

En **IConverterHelper** agrego:

UserResponse ToUserResponse(User user);

Y la implementación en **ConverterHelper**

public UserResponse ToUserResponse(User user)

{

return new UserResponse

{

Address = user.Address,

Document = user.Document,

DocumentTypeId = user.DocumentType.Id,

Email = user.Email,

FirstName = user.FirstName,

Id = user.Id,

ImageId = user.ImageId,

LastName = user.LastName,

PhoneNumber = user.PhoneNumber,

UserType = user.UserType,

};

}

En **AccountController** modifico:

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Tokens;

using System;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using System.Threading.Tasks;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.Api.Models;

namespace Vehicles.API.Controllers.API

{

[ApiController]

[Route("api/[controller]")]

public class AccountController : ControllerBase

{

private readonly IUserHelper \_userHelper;

private readonly IConfiguration \_configuration;

private readonly IConverterHelper \_converterHelper;

public AccountController(IUserHelper userHelper, IConfiguration configuration,IConverterHelper converterHelper)

{

\_userHelper = userHelper;

\_configuration = configuration;

\_converterHelper = converterHelper;

}

[HttpPost]

[Route("CreateToken")]

public async Task<IActionResult> CreateToken([FromBody] LoginViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userHelper.GetUserAsync(model.Username);

UserResponse user2 = \_converterHelper.ToUserResponse(user);

if (user != null)

{

var result = await \_userHelper.ValidatePasswordAsync(user, model.Password);

if (result.Succeeded)

{

Claim[] claims = new[]

{

new Claim(JwtRegisteredClaimNames.Sub, user.Email),

new Claim(JwtRegisteredClaimNames.Jti, Guid.NewGuid().ToString())

};

SymmetricSecurityKey key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(\_configuration["Tokens:Key"]));

SigningCredentials credentials = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

JwtSecurityToken token = new JwtSecurityToken(

\_configuration["Tokens:Issuer"],

\_configuration["Tokens:Audience"],

claims,

expires: DateTime.UtcNow.AddDays(99),

signingCredentials: credentials);

var results = new

{

token = new JwtSecurityTokenHandler().WriteToken(token),

expiration = token.ValidTo,

user2

};

return Created(string.Empty, results);

}

}

}

return BadRequest();

}

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

En el Postman hacemos:



Copiamos el Token y en las otras operaciones de Postman hacemos:



# 42 – Publicar Api

Publicamos el API y probamos con Postman.

No olvidar ANTES poner las direcciones web correctas de las imágenes

# 43 - Flutter

Utilizaremos **dartpad**

<https://dartpad.dev/?null_safety=true>

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  for (int i = 0; i < 5; i++) {  print('hello ${i + 1}');  }  } | Console  hello 1  hello 2  hello 3  hello 4  hello 5 |

## Hola mundo!!

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  print ('Hello world!!');  } | Console  Hello world!! |

## Tipos de datos

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  String name="Luis";  int counter =5;  double temp = 24.5;  bool sw=true;  int? counter2 =null;  double? temp2 = null;  bool? sw2=null;  DateTime born=DateTime(1965,6,23);  print ('Hello $name!!');  print ('$counter,$temp,$sw,$counter2,$temp2,$sw2');  print('Primera letra "${name[0]}"');  print('Ultima letra "${name[name.length-1]}"');  print('Fecha: $born');  } | Console  Hello Luis!!  5,24.5,true,null,null,null  Primera letra "L"  Ultima letra "s"  Fecha: 1965-06-23 00:00:00.000  Con el ? se hacen variables nullables |

## Condiciones

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  bool sw=true;  if(!sw)  {  print('Es verdadero');  }  else  {  print('Es falso');  };    sw ? print('Es verdadero'):print('Es falso');  } | Console  Es falso  Es verdadero  Operador ternario |

## Arreglos

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  List<int> numbers = [1,2,3,4,5];  print (numbers);  print (numbers[1]);  numbers.add(6);  print (numbers);  } | Console  [1, 2, 3, 4, 5]  2  [1, 2, 3, 4, 5, 6] |

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  final numbers = List.generate(15,(int index)=>index);  print (numbers);  } | Console  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14] |

## Diccionario de Datos (Maps)

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  Map<String,dynamic> person = {  'firstname':'Luis',  'lastname':'Nuñez',  'children':0,  'single':true,  'born':DateTime(1965,6,23)  };  print(person);  person.addAll({'nickName':'Lucho'});  print(person);  } | Console  {firstname: Luis, lastname: Nuñez, children: 0, single: true, born: 1965-06-23 00:00:00.000}  {firstname: Luis, lastname: Nuñez, children: 0, single: true, born: 1965-06-23 00:00:00.000, nickName: Lucho} |

## Clases, propiedades y métodos

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  Employee employee = Employee(name:'Pedro',salary:140000);  print (employee);  }  class Employee {  String name;  double salary;    Employee({required this.name,required this.salary});    String toString() {  return '$name => \$$salary';  }  } | Console  Pedro => $140000  Instanciar una clase  Definición de propiedades de la Clase Employee  Constructor  Overrite del método toString |

## Getter y Setter

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  Square c1 = new Square(l:10) ;  print (c1.area);  c1.lado=12;  print (c1.area);  }  class Square {  double l;    Square({required this.l});    double get area => l\*l;    set lado(double value) {  this.l =value;  }  } | Console  100  144  Establecemos un get  Y establecemos un set |

## Clase abstracta

En una clase abstracta se puede o no definir métodos

No se pueden instanciar objetos de clases abstractas

Se crean clases comunes que implementan las Clases Abstractas

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  Dog milu =Dog(patas:4);  Roast claudio =Roast(patas:2);  milu.sound();  claudio.sound();  }  abstract class Animal {  int? patas;  void sound();  }  class Dog implements Animal{  int? patas;  Dog({this.patas});  void sound() => print("Guauuuu, Patas: $patas");  }  class Roast implements Animal{  int? patas;  Roast({this.patas});  void sound() => print("Ki ki ri kiii, Patas: $patas");  } | Guauuuu, Patas: 4  Ki ki ri kiii, Patas: 2  Clase abstracta  Clase que implementa la clase abstracta  Otra clase que implementa la clase abstracta |

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  var eh=HoursEmployee(hours:12,value:7.89);  var ea=SalaryEmployee(salary:720);  print('Por horas: ${eh.getSalary()}');  print('Por salario: ${ea.getSalary()}');  }  abstract class Employee {  double getSalary();  }  class HoursEmployee implements Employee{  int hours;  double value;  HoursEmployee({this.hours=0,this.value=0});  double getSalary(){  return hours \* value;  }  }  class SalaryEmployee implements Employee{  double salary;  SalaryEmployee({this.salary=0});  double getSalary(){  return salary\*0.9;  }  } | Console  Por horas: 94.67999999999999  Por salario: 648  Clase abstracta  Clase que implementa la clase abstracta  Otra clase que implementa la clase abstracta |

## Herencia

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  Trip v1 = Trip(id:1, title:"Amazonas",price:850000);  print (v1);  FullTrip v2 = FullTrip(id:2, title:"Guajira",price:950000,data:"Incluye tours");  print (v2);  }  class Trip{  final int id;  final String title;  final double price;    Trip({required this.id,required this.title,required this.price});  String toString() {  return '$id $title $price';  }  }  class FullTrip extends Trip{  final String data;    FullTrip({required this.data, required int id, required String title, required double price}):super(id:id,title:title,price:price);  String toString() {  return '$id $title $price $data';  }  } | Console  1 Amazonas 850000  2 Guajira 950000 Incluye tours  Clase original  Clase con Herencia |

## Mixings

Dart permite heredar de varias clases



|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  final fliper =Delfin();  fliper.nadar();  final donald=Pato();  donald.caminar();  donald.nadar();  donald.volar();  }  abstract class Animal{}    abstract class Mamifero extends Animal{}  abstract class Ave extends Animal{}  abstract class Pez extends Animal{}  abstract class Volador{  void volar()=>print("Puedo volar!");  }  abstract class Nadador{  void nadar()=>print("Puedo nadar!");  }  abstract class Caminante{  void caminar()=>print("Puedo caminar!");  }  class Delfin extends Mamifero with Nadador{}  class Murcielago extends Mamifero with Caminante, Volador{}  class Gato extends Mamifero with Caminante{}  class Paloma extends Ave with Caminante,Volador{}  class Pato extends Ave with Caminante,Volador,Nadador{}  class Tiburon extends Pez with Nadador{}  class PezVolador extends Pez with Nadador,Volador{} | Console  Puedo nadar!  Puedo caminar!  Puedo nadar!  Puedo volar! |

## Futures

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() {  print('Antes de la petición');  httpGet('https://zulu.net/api/data').then((data){  print(data.toUpperCase());  });    print('Después de la petición');    }  Future<String> httpGet(String url){  return Future.delayed(  Duration(seconds:3),() => 'Hola mundo depues de 3 segundos'  );  } | Console  Antes de la petición  Después de la petición  HOLA MUNDO DESPUES DE 3 SEGUNDOS |

|  |  |
| --- | --- |
| **Código** | **Vista y comentario** |
| void main() async {  print('Antes de la petición');  String data = await httpGet('https://zulu.net/api/data');  print(data.toUpperCase());  print('Después de la petición');    }  Future<String> httpGet(String url){  return Future.delayed(  Duration(seconds:3),() {  return 'Hola mundo esperé 3 segundos';  }  );  } | Console  Antes de la petición  HOLA MUNDO ESPERÉ 3 SEGUNDOS  Después de la petición |

## Instalación de Ambiente de Desarrollo para Flutter

**Instalación Herramientas**

* Visual Studio Code, <https://code.visualstudio.com/>
* GIT, <https://git-scm.com/>
* Postman, <https://www.postman.com/downloads/>
* Android Studio, <https://developer.android.com/studio>
* Git Hub Desktop: <https://desktop.github.com/>
* XCode, <https://apps.apple.com/hn/app/xcode/id497799835?l=en&mt=12> (Solo Mac)

**Plugins Visual Studio Code**

* Awesome Flutter Snippets
* Bracket Pair Colorizer 2
* Dart
* Flutter
* Material Icon Theme
* Paste JSON as Code
* Terminal
* Auto Close Tags
* Beautify

**Instalación del SDK de Flutter**

<https://flutter.dev/>

1. Descargamos el Flutter SDK
2. Descomprimirlo en la carpeta **c:\src\flutter**
3. Agregar en las variables de entorno y agregamos a la variable **path** la carpeta: **c:\src\flutter\bin**
4. Verificamos abriendo un **Git Bash** y corriendo el comando: **flutter --version**
5. Para verificar que todo está bien corremos el comando: **flutter doctor**
6. Le agregamos el plugin de **Flutter** al Android Studio.
7. Debemos de aceptar las licencias de Android, para eso abrimos un Git Bash, con el comando: **flutter doctor --android-licenses**
8. Verificamos de nuevo con el flutter doctor
9. Creamos un proyecto Flutter en Android Studio
10. Luego creamos un Android Virtual por Tools / AVD Manager.
11. Corremos la aplicación de prueba.
12. Ahora vamos a correr Flutter desde el Visual Studio Code.
13. Corremos el **View** / **Command Palette** (Ctrl + Shift + P)
14. Escribimos **Flutter** y seleccionamos: **New Application Flutter**
15. Luego corremos nuevamente el **View** / **Command Palette** (Ctrl + Shift + P)
16. Escribimos **Flutter** y seleccionamos: **Launch Emulator** y seleccionamos uno de los emuladores previamente creados.
17. Luego presionamos F5 para ejecutar la aplicación.
18. Probemos el hot reload.

## Conceptos básicos de Flutter

* **Widget**: clase con argumentos posicionales o argumentos con nombre.
* **Stateless Widget:** sin estado, no pueden cambiar las propiedades.
* **Stateful Widget:** con estado, las propiedades si cambian. Puede re dibujarse a sí mismo.
* **Árbol de Widgets:** cómo se organizan los Widgets.

# Primera App: Contador de Taps

Toda app en Flutter empieza con un **main**:

void main(){

}

El primer comando es **RunApp**:

import 'package:flutter/material.dart';

void main(){

  runApp(app)

}

Cupertino: es para dar una apariencia iOS

Material: es para dar una apariencia Android

Borramos la carpeta **test**

Dentro de la carpeta **lib** creamos el archivo **app.dart**

Snnipet **importM** crea la línea

import 'package:flutter/material.dart';

Snnipet **stl** crea el código

class  extends StatelessWidget {

  const ({ Key? key }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Container(

    );

  }

}

Le ponemos nombre **MyApp**

import 'package:flutter/material.dart';

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Container();

  }

}

Cambiamos **Container** por **Material**

import 'package:flutter/material.dart';

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Center(

        child: Text('Hola mundo desde cero'),

      ),

    );

  }

}

Y en **main.dart** hacemos:

import 'package:flutter/material.dart';

import 'package:my\_second\_flutter\_app/app.dart';

void main() {

  runApp(MyApp());

}

Corremos la aplicación y tenemos:



Dentro de la carpeta **lib** creamos el archivo **counter.dart**

En **app.dart** reemplazamos el **Text**

Usamos el Snnipet **importM** para poner la línea de **material**, y el Snnipet **stlf** para poner un widget con estado

Le ponemos Counter de nombre:

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({ Key? key }) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  @override

  Widget build(BuildContext context) {

    return Container(

    );

  }

}

Esto retorna un **Container**. Dentro del **Container** ponemos:

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({Key? key}) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  @override

  Widget build(BuildContext context) {

    return Container(

      child: Center(

        child: Text('Hola de nuevo'),

      ),

    );

  }

}

Y en **app.dart** cambiamos, en vez del **Text** ponemos la página **Counter**:

import 'package:flutter/material.dart';

import 'package:my\_second\_flutter\_app/counter.dart';

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      home: Center(

        child: Counter(),

      ),

    );

  }

}

Ahora en la página **counter.dart** cambiamos el **Container** por un **Scaffold**

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({Key? key}) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Mi segunda App')),

      body: Center(

        child: Text('Hola mundo pero mas bonito'),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {},

      ),

    );

  }

}

Entonces la App se ve así:

Captura de pantalla de un celular

Descripción generada automáticamente

Los íconos los podemos ver en la página

<https://fonts.google.com/icons>

La rayita de arriba que dice Debug se saca poniendo en **app.dart**:

import 'package:flutter/material.dart';

import 'package:my\_second\_flutter\_app/counter.dart';

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      home: Center(

        child: Counter(),

      ),

    );

  }

}

Cambiamos el testo, y le agrandamos el tamaño:

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({Key? key}) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Mi segunda App')),

      body: Center(

        child: Text(

          'Contador de taps',

          style: TextStyle(fontSize: 20),

        ),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {},

      ),

    );

  }

}

Para poner otro texto hacemos Control Punto en **Text** y elegimos **Wrap with Column**

Copiamos y pegamos el Text para que queden los dos Text como children del Colum

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({Key? key}) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Mi segunda App')),

      body: Center(

        child: Column(

          children: [

            Text(

              'Contador de taps',

              style: TextStyle(fontSize: 20),

            ),

            Text(

              'Contador de taps',

              style: TextStyle(fontSize: 20),

            ),

          ],

        ),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {},

      ),

    );

  }

}

Pero quedó mal alineación:

Captura de pantalla de un celular

Descripción generada automáticamente

Hacemos lo siguiente:

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({Key? key}) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Mi segunda App')),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            Text(

              'Contador de taps',

              style: TextStyle(fontSize: 20),

            ),

            Text(

              '0',

              style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold),

            ),

          ],

        ),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {},

      ),

    );

  }

}

Y ahora se ve así:

Captura de pantalla de un celular

Descripción generada automáticamente

Creamos la propiedad privada **\_counter**

import 'package:flutter/material.dart';

class Counter extends StatefulWidget {

  const Counter({Key? key}) : super(key: key);

  @override

  \_CounterState createState() => \_CounterState();

}

class \_CounterState extends State<Counter> {

  int \_counter = 0;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: Text('Mi segunda App')),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            Text(

              'Contador de taps',

              style: TextStyle(fontSize: 20),

            ),

            Text(

              '$\_counter',

              style: TextStyle(fontSize: 40, fontWeight: FontWeight.bold),

            ),

          ],

        ),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {

          setState(() {

            \_counter++;

          });

        },

      ),

    );

  }

}

Y tenemos el contador de taps funcionando

# Creación del Proyecto vehicles\_app

Creamos el Proyecto **vehicles\_app**

Borramos la carpeta **test**

Borramos el contenido de **main.dart**

Corremos el Snnipet **mateapp** (nos hace un Hola Mundo Android)

Con el Snnipet **cupeapp** nos hace un Hola Mundo iOS)

# Guardar el Proyecto en el Repositorio

En GitHub creo el repositorio VehiclesApp (sin gitignore)

Con el explorador de archivos vamos ala carpeta **F:\Flutter\vehicles\_app**

Hacemos clic derecho **Git Bash Here**

Escribimos el comando

git init

git add .

git commit -m "Basic Project creation"

git remote add origin https://github.com/LuisPulenta/VehiclesApp.git

git push origin master

Luego abrimos el **GitHub DeskTop** y elegimos **File – Add local Repository** y buscamos la carpeta **F:\Flutter\vehicles\_app**

Y nos queda listo para hacer los Push desde aquí

# Login – Diseño de Pantalla

En la carpeta **F:\Flutter\vehicles\_app** creamos la carpeta **assets**

Dentro ponemos los archivos **logo.png** y **noimage.png**

Vamos al archivo **pubspec.yaml** y ahí hacemos:

# To add assets to your application, add an assets section, like this:

  assets:

     - assets/

  #   - images/a\_dot\_ham.jpeg

Dentro de la carpeta **lib** creamos una carpeta que llamamos **screens** y dentro creamos el archivo **login\_screen.dart**

Con **stf** creamos un widget con estado y le ponemos como nombre **LoginScree.**

Reemplazamos el **Container** por un **Scaffold** con un **Text** que diga **Login Screen**:

import 'package:flutter/material.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({Key? key}) : super(key: key);

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Text('Login Screen'),

    );

  }

}

En **main.dart** cambiamos el **title** por **Vehicles App** y el **home** que devuelva una **LoginScreen**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/screens/login\_screen.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Vehicles App',

      home: LoginScreen(),

    );

  }

}

En **login\_screen.dart** hacemos:

import 'package:flutter/material.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({Key? key}) : super(key: key);

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  String \_email = '';

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Center(

          child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          \_showLogo(),

          \_showEmail(),

        ],

      )),

    );

  }

  Widget \_showLogo() {

    return Image(

      image: AssetImage('assets/logo.png'),

      width: 300,

    );

  }

  Widget  \_showEmail() {

    return Container(

      padding: EdgeInsets.all(20),

      child: TextField(

        autofocus: true,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa tu Email...',

            labelText: 'Email',

            suffixIcon: Icon(Icons.email),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_email = value;

          print(\_email);

        },

      ),

    );

  }

}

Finalmente agregamos Password, Rememberme y los botones:

import 'package:flutter/material.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({Key? key}) : super(key: key);

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  String \_email = '';

  String \_password = '';

  bool \_rememberme = true;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Center(

          child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          \_showLogo(),

          SizedBox(

            height: 20,

          ),

          \_showEmail(),

          \_showPassword(),

          \_showRememberme(),

          \_showButtons(),

        ],

      )),

    );

  }

  Widget \_showLogo() {

    return Image(

      image: AssetImage('assets/logo.png'),

      width: 300,

    );

  }

  Widget \_showEmail() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa tu Email...',

            labelText: 'Email',

            suffixIcon: Icon(Icons.email),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_email = value;

        },

      ),

    );

  }

  Widget \_showPassword() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: true,

        decoration: InputDecoration(

            hintText: 'Ingresa tu Contraseña...',

            labelText: 'Contraseña',

            suffixIcon: Icon(Icons.lock),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_password = value;

        },

      ),

    );

  }

  \_showRememberme() {

    return CheckboxListTile(

      title: Text('Recordarme:'),

      value: \_rememberme,

      onChanged: (value) {

        setState(() {

          \_rememberme = value!;

        });

      },

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Iniciar Sesión'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () {},

            ),

          ),

          SizedBox(

            width: 20,

          ),

          Expanded(

            child: ElevatedButton(

              child: Text('Nuevo Usuario'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Colors.purple;

                }),

              ),

              onPressed: () {},

            ),

          ),

        ],

      ),

    );

  }

}

Hay dos formas de poner color:

return Color(0xFF120E43)

return Colors.purple

En esta página se puede buscar colores:

<https://uicolorpicker.com/>

# Login – Métodos

## Instalación de paquete validador de Email

Vamos a la página

<https://pub.dev/>

Buscamos **email\_validator**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

dependencies:

email\_validator: ^2.0.1

Copiamos la línea

email\_validator: ^2.0.1

y la pegamos en el archivo **pubspec.yaml**

dependencies:

  flutter:

    sdk: flutter

  # The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.2

  email\_validator: ^2.0.1

## Validación de Email y Password

import 'package:email\_validator/email\_validator.dart';

import 'package:flutter/material.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({Key? key}) : super(key: key);

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  String \_email = '';

  String \_emailError = '';

  bool \_emailShowError = false;

  String \_password = '';

  String \_passwordError = '';

  bool \_passwordShowError = false;

  bool \_rememberme = true;

  bool \_passwordShow = false;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Center(

          child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: <Widget>[

          \_showLogo(),

          SizedBox(

            height: 20,

          ),

          \_showEmail(),

          \_showPassword(),

          \_showRememberme(),

          \_showButtons(),

        ],

      )),

    );

  }

  Widget \_showLogo() {

    return Image(

      image: AssetImage('assets/logo.png'),

      width: 300,

    );

  }

  Widget \_showEmail() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa tu Email...',

            labelText: 'Email',

            errorText: \_emailShowError ? \_emailError : null,

            prefixIcon: Icon(Icons.alternate\_email),

            suffixIcon: Icon(Icons.email),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_email = value;

        },

      ),

    );

  }

  Widget \_showPassword() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: !\_passwordShow,

        decoration: InputDecoration(

            hintText: 'Ingresa tu Contraseña...',

            labelText: 'Contraseña',

            errorText: \_passwordShowError ? \_passwordError : null,

            prefixIcon: Icon(Icons.lock),

            suffixIcon: IconButton(

              icon: \_passwordShow

                  ? Icon(Icons.visibility)

                  : Icon(Icons.visibility\_off),

              onPressed: () {

                setState(() {

                  \_passwordShow = !\_passwordShow;

                });

              },

            ),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_password = value;

        },

      ),

    );

  }

  \_showRememberme() {

    return CheckboxListTile(

      title: Text('Recordarme:'),

      value: \_rememberme,

      onChanged: (value) {

        setState(() {

          \_rememberme = value!;

        });

      },

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Iniciar Sesión'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_login(),

            ),

          ),

          SizedBox(

            width: 20,

          ),

          Expanded(

            child: ElevatedButton(

              child: Text('Nuevo Usuario'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Colors.purple;

                }),

              ),

              onPressed: () {},

            ),

          ),

        ],

      ),

    );

  }

  void \_login() {

    if (!validateFields()) {

      return;

    }

  }

  bool validateFields() {

    bool isValid = true;

    if (\_email.isEmpty) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar tu Email';

    } else if (!EmailValidator.validate(\_email)) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar un Email válido';

    } else {

      \_emailShowError = false;

    }

    if (\_password.isEmpty) {

      isValid = false;

      \_passwordShowError = true;

      \_passwordError = 'Debes ingresar tu Contraseña';

    } else if (\_password.length < 6) {

      isValid = false;

      \_passwordShowError = true;

      \_passwordError = 'La Contraseña debe tener al menos 6 caracteres';

    } else {

      \_passwordShowError = false;

    }

    setState(() {});

    return isValid;

  }

}

# Conexión y uso del API

Instalamos la extensión **Thunder Client**

Una vez instalado nos aparece un rayito a la izquierda

Lo elegimos. Ponemos New Request. Elegimos POST y ponemos la dirección Web del Api.

Elegimos Body, Json y escribimos el Request.

Le damos Send y ibtenemos la respuesta:



Copiamos la respuesta y vamos a la página

<https://javiercbk.github.io/json_to_dart/>

Allí pegamos , ponemos el nombre y generamos la Clase en Dart:



Dentro de la carpeta **lib** creamos una carpeta que llamamos **models** y dentro creamos el archivo **document\_type.dart**

class DocumentType {

  int id = 0;

  String description = '';

  DocumentType({required this.id, required this.description});

  DocumentType.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    description = json['description'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['description'] = this.description;

    return data;

  }

}

Creamos la clase **userr2.dart**

class User2 {

  String firstName = '';

  String lastName = '';

  String document = '';

  String address = '';

  String imageId = '';

  String phoneNumber = '';

  String email = '';

  String id = '';

  int documentTypeId = 0;

  String imageFullPath = '';

  int userType = 0;

  String fullName = '';

  User2(

      {required this.firstName,

      required this.lastName,

      required this.document,

      required this.address,

      required this.imageId,

      required this.phoneNumber,

      required this.email,

      required this.id,

      required this.documentTypeId,

      required this.imageFullPath,

      required this.userType,

      required this.fullName});

  User2.fromJson(Map<String, dynamic> json) {

    firstName = json['firstName'];

    lastName = json['lastName'];

    document = json['document'];

    address = json['address'];

    imageId = json['imageId'];

    phoneNumber = json['phoneNumber'];

    email = json['email'];

    id = json['id'];

    documentTypeId = json['documentTypeId'];

    imageFullPath = json['imageFullPath'];

    userType = json['userType'];

    fullName = json['fullName'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['firstName'] = this.firstName;

    data['lastName'] = this.lastName;

    data['document'] = this.document;

    data['address'] = this.address;

    data['imageId'] = this.imageId;

    data['phoneNumber'] = this.phoneNumber;

    data['email'] = this.email;

    data['id'] = this.id;

    data['documentTypeId'] = this.documentTypeId;

    data['imageFullPath'] = this.imageFullPath;

    data['userType'] = this.userType;

    data['fullName'] = this.fullName;

    return data;

  }

}

Creamos la clase **token.dart**

import 'package:vehicles\_app/models/user2.dart';

class Token {

  String token = '';

  String expiration = '';

  User2 user2 = User2(

      firstName: '',

      lastName: '',

      document: '',

      address: '',

      imageId: '',

      phoneNumber: '',

      email: '',

      id: '',

      documentTypeId: 0,

      imageFullPath: '',

      userType: 0,

      fullName: '');

  Token({required this.token, required this.expiration, required this.user2});

  Token.fromJson(Map<String, dynamic> json) {

    token = json['token'];

    expiration = json['expiration'];

    user2 = User2.fromJson(json['user2']);

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['token'] = this.token;

    data['expiration'] = this.expiration;

    data['user2'] = this.user2.toJson();

    return data;

  }

}

Creamos dentro de **lib** una carpeta que llamamos **helpers** y dentro creamos el archivo **constants.dart**

class Constants {

  static String get apiUrl => 'http://keypress.serveftp.net:88/VehiclesApi';

}

Vamos a la página

<https://pub.dev/>

Buscamos **http**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

dependencies:

http: ^0.13.3

Copiamos la línea

http: ^0.13.3

y la pegamos en el archivo **pubspec.yaml**

# The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.2

  email\_validator: ^2.0.1

  http: ^0.13.3

En **login\_screen.dart** hacemos:

Arriba:

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:email\_validator/email\_validator.dart';

import 'package:http/http.dart' as http;

import 'package:vehicles\_app/helpers/constants.dart';

y en el método login:

void \_login() async {

    setState(() {

      \_passwordShow = false;

    });

    if (!validateFields()) {

      return;

    }

    Map<String, dynamic> request = {

      'userName': \_email,

      'password': \_password,

    };

    var url = Uri.parse('${Constants.apiUrl}/api/Account/CreateToken');

    var response = await http.post(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

      },

      body: jsonEncode(request),

    );

    print(response.body);

  }

# Loader

Creamos en la carpeta **lib** una carpeta que llamamos **components y** dentro creamos un archivo que llamamos **loader\_component.dart**

import 'package:flutter/material.dart';

class LoaderComponent extends StatelessWidget {

  final String text;

  LoaderComponent({this.text = ''});

  @override

  Widget build(BuildContext context) {

    return Center(

      child: Container(

        width: 200,

        height: 100,

        decoration: BoxDecoration(

            color: Colors.yellow[300], borderRadius: BorderRadius.circular(10)),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            CircularProgressIndicator(),

            SizedBox(

              height: 20,

            ),

            Text(text, style: TextStyle(fontSize: 20)),

          ],

        ),

      ),

    );

  }

}

En **login\_screen.dart** hacemos:

Arriba:

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:email\_validator/email\_validator.dart';

import 'package:http/http.dart' as http;

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/token.dart';

class LoginScreen extends StatefulWidget {

  const LoginScreen({Key? key}) : super(key: key);

  @override

  \_LoginScreenState createState() => \_LoginScreenState();

}

class \_LoginScreenState extends State<LoginScreen> {

  String \_email = '';

  String \_emailError = '';

  bool \_emailShowError = false;

  String \_password = '';

  String \_passwordError = '';

  bool \_passwordShowError = false;

  bool \_rememberme = true;

  bool \_passwordShow = false;

  bool \_showLoader = false;

  @override

Widget build(BuildContext context) {

    return Scaffold(

      body: Stack(

        children: <Widget>[

          Column(

            mainAxisAlignment: MainAxisAlignment.center,

            children: <Widget>[

              \_showLogo(),

              SizedBox(

                height: 20,

              ),

              \_showEmail(),

              \_showPassword(),

              \_showRememberme(),

              \_showButtons(),

            ],

          ),

          \_showLoader

              ? LoaderComponent(

                  text: 'Por favor espere...',

                )

              : Container(),

        ],

      ),

    );

  }

Y en el método login:

void \_login() async {

    setState(() {

      \_passwordShow = false;

    });

    if (!validateFields()) {

      return;

    }

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'userName': \_email,

      'password': \_password,

    };

    var url = Uri.parse('${Constants.apiUrl}/api/Account/CreateToken');

    var response = await http.post(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

      },

      body: jsonEncode(request),

    );

    setState(() {

      \_showLoader = false;

    });

    if (response.statusCode >= 400) {

      setState(() {

        \_emailShowError = true;

        \_emailError = 'Email o Contraseña no válidos';

        \_passwordShowError = true;

        \_passwordError = 'Email o Contraseña no válidos';

      });

      return;

    }

    var body = response.body;

    var decodedJson = jsonDecode(body);

    var token = Token.fromJson(decodedJson);

    print(token.token);

  }

# Pasar a otra pantalla desde el Login

En la carpeta **lib/screens** creamos el archivo **home\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/login\_screen.dart';

class HomeScreen extends StatefulWidget {

  final Token token;

  HomeScreen({required this.token});

  @override

  \_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Vehicles'),

      ),

      body: \_getBody(),

      drawer: \_getMechanicMenu(),

    );

  }

  Widget \_getBody() {

    return Container(

      margin: EdgeInsets.all(30),

      child: Center(

        child: Text(

          'Bienvenido/a ${widget.token.user2.fullName}',

          style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getMechanicMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Marcas'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.precision\_manufacturing),

            title: Text('Procedimientos'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.badge),

            title: Text('Tipos de Documento'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.toys),

            title: Text('Tipos de Vehículo'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.people),

            title: Text('Usuarios'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

            },

          ),

        ],

      ),

    );

  }

}

En **login\_screen.dart** hacemos en el método **login**:

void \_login() async {

    setState(() {

      \_passwordShow = false;

    });

    if (!validateFields()) {

      return;

    }

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'userName': \_email,

      'password': \_password,

    };

    var url = Uri.parse('${Constants.apiUrl}/api/Account/CreateToken');

    var response = await http.post(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

      },

      body: jsonEncode(request),

    );

    setState(() {

      \_showLoader = false;

    });

    if (response.statusCode >= 400) {

      setState(() {

        \_emailShowError = true;

        \_emailError = 'Email o Contraseña no válidos';

        \_passwordShowError = true;

        \_passwordError = 'Email o Contraseña no válidos';

      });

      return;

    }

    var body = response.body;

    var decodedJson = jsonDecode(body);

    var token = Token.fromJson(decodedJson);

    Navigator.pushReplacement(

        context,

        MaterialPageRoute(

            builder: (context) => HomeScreen(

                  token: token,

                )));

  }

# Distintos Menús para Mecánico o Cliente

En **home\_screen.dart** hacemos:

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/login\_screen.dart';

class HomeScreen extends StatefulWidget {

  final Token token;

  HomeScreen({required this.token});

  @override

  \_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Vehicles'),

      ),

      body: \_getBody(),

      drawer: widget.token.user2.userType == 0

          ? \_getMechanicMenu()

          : \_getCustomerMenu(),

    );

  }

  Widget \_getBody() {

    return Container(

      margin: EdgeInsets.all(30),

      child: Center(

        child: Text(

          'Bienvenido/a ${widget.token.user2.fullName}',

          style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getMechanicMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Marcas'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.precision\_manufacturing),

            title: Text('Procedimientos'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.badge),

            title: Text('Tipos de Documento'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.toys),

            title: Text('Tipos de Vehículo'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.people),

            title: Text('Usuarios'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

            },

          ),

        ],

      ),

    );

  }

  \_getCustomerMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Mis Vehículos'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

            },

          ),

        ],

      ),

    );

  }

}

# Agregar foto a home\_screen.dart

En **home\_screen.dart** hacemos en el método **\_getbody**:

Widget \_getBody() {

    return Container(

      margin: EdgeInsets.all(30),

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: [

          ClipRRect(

            borderRadius: BorderRadius.circular(150),

            child: FadeInImage(

                placeholder: AssetImage('assets/logo.png'),

                image: NetworkImage(widget.token.user2.imageFullPath),

                height: 300,

                fit: BoxFit.cover),

          ),

          SizedBox(

            height: 30,

          ),

          Center(

            child: Text(

              'Bienvenid@ ${widget.token.user2.fullName}',

              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

            ),

          )

        ],

      ),

    );

  }

# Obtener Procedures desde el API

Obtenemos la respuesta del Get Procedures



La copiamos y la pegamos en la página JSON to Dart



Copiamos y luego en la carpeta **models** creamos el archivo **procedure.dart** y pegamos.

Y corregimos algunas cosas:

class Procedure {

  int id=0;

  String description='';

  double price=0;

  Procedure({required this.id,required  this.description,required  this.price});

  Procedure.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    description = json['description'];

    price = json['price'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['description'] = this.description;

    data['price'] = this.price;

    return data;

  }

}

En la carpeta **screens** creamos el archivo **procedures\_screen.dart**

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/token.dart';

class ProceduresScreen extends StatefulWidget {

  final Token token;

  ProceduresScreen({required this.token});

  @override

  \_ProceduresScreenState createState() => \_ProceduresScreenState();

}

class \_ProceduresScreenState extends State<ProceduresScreen> {

  List<Procedure> \_procedures = [];

  @override

  void initState() {

    super.initState();

    \_getProcedures();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Procedimientos'),

      ),

      body: Center(child: Text('Procedimientos'),)

    );

  }

  void \_getProcedures() async {

    var url = Uri.parse('${Constants.apiUrl}/api/Procedures');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${widget.token.token}',

      },

    );

    print(response.body);

  }

}

Modificamos **home\_screen.dart**

Widget \_getMechanicMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Marcas'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.precision\_manufacturing),

            title: Text('Procedimientos'),

            onTap: () {

              Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => ProceduresScreen(

                  token: widget.token,

                )));

            },

          ),

          ListTile(

            leading: Icon(Icons.badge),

            title: Text('Tipos de Documento'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.toys),

            title: Text('Tipos de Vehículo'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.people),

            title: Text('Usuarios'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

            },

          ),

        ],

      ),

    );

  }

Ponemos el componente **Loader** al **procedures\_screen**

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/token.dart';

class ProceduresScreen extends StatefulWidget {

  final Token token;

  ProceduresScreen({required this.token});

  @override

  \_ProceduresScreenState createState() => \_ProceduresScreenState();

}

class \_ProceduresScreenState extends State<ProceduresScreen> {

  List<Procedure> \_procedures = [];

  bool \_showLoader = false;

  @override

  void initState() {

    super.initState();

    \_getProcedures();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text('Procedimientos'),

        ),

        body: Center(

          child: \_showLoader

              ? LoaderComponent(text: 'Por favor espere...')

              : Text('Procedimientos'),

        ));

  }

  void \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    var url = Uri.parse('${Constants.apiUrl}/api/Procedures');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${widget.token.token}',

      },

    );

    setState(() {

      \_showLoader = false;

    });

    print(response.body);

  }

}

Modificamos el método **\_getprocedures**

void \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    var url = Uri.parse('${Constants.apiUrl}/api/Procedures');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${widget.token.token}',

      },

    );

    setState(() {

      \_showLoader = false;

    });

    var body = response.body;

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        \_procedures.add(Procedure.fromJson(item));

      }

    }

    print(\_procedures);

  }

# Pintar los Procedures

Para ver los precios en formato moneda debemos agregar un paquete.

Vamos a la página <https://pub.dev/> y buscamos **intl**

Para instalarlo debemos hacer;

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

dependencies:

intl: ^0.17.0

En **procedures\_screen.dart** hacemos:

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/token.dart';

class ProceduresScreen extends StatefulWidget {

  final Token token;

  ProceduresScreen({required this.token});

  @override

  \_ProceduresScreenState createState() => \_ProceduresScreenState();

}

class \_ProceduresScreenState extends State<ProceduresScreen> {

  List<Procedure> \_procedures = [];

  bool \_showLoader = false;

  @override

  void initState() {

    super.initState();

    \_getProcedures();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Procedimientos'),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(text: 'Por favor espere...')

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {},

      ),

    );

  }

  void \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    var url = Uri.parse('${Constants.apiUrl}/api/Procedures');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${widget.token.token}',

      },

    );

    setState(() {

      \_showLoader = false;

    });

    var body = response.body;

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        \_procedures.add(Procedure.fromJson(item));

      }

    }

    print(\_procedures);

  }

  Widget \_getContent() {

    return \_procedures.length == 0 ? \_noContent() : \_getListView();

  }

  Widget \_noContent() {

    return Container(

      margin: EdgeInsets.all(20),

      child: Center(

        child: Text(

          'No hay procedimientos almacenados',

          style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getListView() {

    return ListView(

      children: \_procedures.map((e) {

        return Card(

          child: InkWell(

            onTap: () {},

            child: Container(

              margin: EdgeInsets.all(10),

              padding: EdgeInsets.all(5),

              child: Column(

                children: [

                  Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: [

                      Text(

                        e.description,

                        style: TextStyle(fontSize: 16),

                      ),

                      Icon(Icons.arrow\_forward\_ios),

                    ],

                  ),

                  SizedBox(

                    height: 15,

                  ),

                  Row(

                    children: [

                      Text(

                        '${NumberFormat.currency(symbol: '\$').format(e.price)}',

                        style: TextStyle(fontWeight: FontWeight.bold),

                      ),

                    ],

                  ),

                ],

              ),

            ),

          ),

        );

      }).toList(),

    );

  }

}

# Creación y Edición de Procedures

En la carpeta **screens** creamos el archivo **procedure\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/token.dart';

class ProcedureScreen extends StatefulWidget {

  final Token token;

  final Procedure procedure;

  ProcedureScreen({required this.token, required this.procedure});

  @override

  \_ProcedureScreenState createState() => \_ProcedureScreenState();

}

class \_ProcedureScreenState extends State<ProcedureScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(widget.procedure.id == 0

            ? 'Nuevo procedimiento'

            : widget.procedure.description),

      ),

      body:Center(child: Text(widget.procedure.id == 0

            ? 'Nuevo procedimiento'

            : widget.procedure.description),)

    );

  }

}

Modificamos **procedures\_screen.dart**

……………..

 Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Procedimientos'),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(text: 'Por favor espere...')

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {

          Navigator.push(

              context,

              MaterialPageRoute(

                  builder: (context) => ProcedureScreen(

                      token: widget.token,

                      procedure: Procedure(description: '', id: 0, price: 0))));

        },

      ),

    );

  }

……………..

Widget \_getListView() {

    return ListView(

      children: \_procedures.map((e) {

        return Card(

          child: InkWell(

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) =>

                          ProcedureScreen(token: widget.token, procedure: e)));

            },

            child: Container(

……………

# Clase genérica para manejar las comunicaciones con el API

En la carpeta **models** creamos un archivo que llamamos **response.dart**

class Response {

  bool isSuccess;

  String message;

  dynamic result;

  Response({required this.isSuccess, this.message = '', this.result});

}

En la carpeta **helpers** creamos **api\_helper.dart**

import 'dart:convert';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:http/http.dart' as http;

import 'constants.dart';

class ApiHelper {

  static Future<Response> getProcedures(String token) async {

    var url = Uri.parse('${Constants.apiUrl}/api/Procedures');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    List<Procedure> list = [];

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        list.add(Procedure.fromJson(item));

      }

    }

    return Response(isSuccess: true, result: list);

  }

}

Vamos a la página <https://pub.dev/> y buscamos **adaptive\_dialog**

Para instalarlo debemos hacer;

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

adaptive\_dialog: ^1.1.0

En **procedures\_screen** modificamos el método **\_getProcedures**

void \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.getProcedures(widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_procedures = response.result;

    });

  }

Agregamos a **api\_helper.dart**

import 'dart:convert';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:http/http.dart' as http;

import 'constants.dart';

class ApiHelper {

  static Future<Response> getProcedures(String token) async {

    var url = Uri.parse('${Constants.apiUrl}/api/Procedures');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    List<Procedure> list = [];

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        list.add(Procedure.fromJson(item));

      }

    }

    return Response(isSuccess: true, result: list);

  }

  static Future<Response> put(String controller, String id,

      Map<String, dynamic> request, String token) async {

    var url = Uri.parse('${Constants.apiUrl}$controller$id');

    var response = await http.put(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

      body: jsonEncode(request),

    );

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: response.body);

    }

    return Response(isSuccess: true);

  }

  static Future<Response> post(

      String controller, Map<String, dynamic> request, String token) async {

    var url = Uri.parse('${Constants.apiUrl}$controller');

    var response = await http.post(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

      body: jsonEncode(request),

    );

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: response.body);

    }

    return Response(isSuccess: true);

  }

  static Future<Response> delete(

      String controller, String id, String token) async {

    var url = Uri.parse('${Constants.apiUrl}$controller$id');

    var response = await http.delete(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: response.body);

    }

    return Response(isSuccess: true);

  }

}

Hacemos en **procedure\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/token.dart';

class ProcedureScreen extends StatefulWidget {

  final Token token;

  final Procedure procedure;

  ProcedureScreen({required this.token, required this.procedure});

  @override

  \_ProcedureScreenState createState() => \_ProcedureScreenState();

}

class \_ProcedureScreenState extends State<ProcedureScreen> {

  String \_description = '';

  String \_descriptionError = '';

  bool \_descriptionShowError = false;

  TextEditingController \_descriptionController = TextEditingController();

  String \_price = '';

  String \_priceError = '';

  bool \_priceShowError = false;

  TextEditingController \_priceController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_description = widget.procedure.description;

    \_descriptionController.text = \_description;

    \_price = widget.procedure.price.toString();

    \_priceController.text = \_price;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(widget.procedure.id == 0

              ? 'Nuevo procedimiento'

              : widget.procedure.description),

        ),

        body: Column(

          children: <Widget>[

            \_showDescription(),

            \_showPrice(),

            \_showButtons(),

          ],

        ));

  }

  Widget \_showDescription() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        controller: \_descriptionController,

        decoration: InputDecoration(

            hintText: 'Ingresa una Descripción...',

            labelText: 'Descripción',

            errorText: \_descriptionShowError ? \_descriptionError : null,

            suffixIcon: Icon(Icons.description),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_description = value;

        },

      ),

    );

  }

  Widget \_showPrice() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType:

            TextInputType.numberWithOptions(decimal: true, signed: false),

        controller: \_priceController,

        decoration: InputDecoration(

            hintText: 'Ingresa un Precio...',

            labelText: 'Precio',

            errorText: \_priceShowError ? \_priceError : null,

            suffixIcon: Icon(Icons.attach\_money),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_price = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => {},

            ),

          ),

          widget.procedure.id == 0

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.procedure.id == 0

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () {},

                  ),

                ),

        ],

      ),

    );

  }

}

# Métodos Post,Put y Delete

En **procedure\_screen.dart** hacemos:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

class ProcedureScreen extends StatefulWidget {

  final Token token;

  final Procedure procedure;

  ProcedureScreen({required this.token, required this.procedure});

  @override

  \_ProcedureScreenState createState() => \_ProcedureScreenState();

}

class \_ProcedureScreenState extends State<ProcedureScreen> {

  bool \_showLoader = false;

  String \_description = '';

  String \_descriptionError = '';

  bool \_descriptionShowError = false;

  TextEditingController \_descriptionController = TextEditingController();

  String \_price = '';

  String \_priceError = '';

  bool \_priceShowError = false;

  TextEditingController \_priceController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_description = widget.procedure.description;

    \_descriptionController.text = \_description;

    \_price = widget.procedure.price.toString();

    \_priceController.text = \_price;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(widget.procedure.id == 0

              ? 'Nuevo procedimiento'

              : widget.procedure.description),

        ),

        body: Stack(

          children: [

            Column(

              children: <Widget>[

                \_showDescription(),

                \_showPrice(),

                \_showButtons(),

              ],

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showDescription() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        controller: \_descriptionController,

        decoration: InputDecoration(

            hintText: 'Ingresa una Descripción...',

            labelText: 'Descripción',

            errorText: \_descriptionShowError ? \_descriptionError : null,

            suffixIcon: Icon(Icons.description),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_description = value;

        },

      ),

    );

  }

  Widget \_showPrice() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType:

            TextInputType.numberWithOptions(decimal: true, signed: false),

        controller: \_priceController,

        decoration: InputDecoration(

            hintText: 'Ingresa un Precio...',

            labelText: 'Precio',

            errorText: \_priceShowError ? \_priceError : null,

            suffixIcon: Icon(Icons.attach\_money),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_price = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.procedure.id == 0

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.procedure.id == 0

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

  void \_save() {

    if (!validateFields()) {

      return;

    }

    widget.procedure.id == 0 ? \_addRecord() : \_saveRecord();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_description.isEmpty) {

      isValid = false;

      \_descriptionShowError = true;

      \_descriptionError = 'Debes ingresar una Descripción';

    } else {

      \_descriptionShowError = false;

    }

    if (\_price.isEmpty) {

      isValid = false;

      \_priceShowError = true;

      \_priceError = 'Debes ingresar un Precio';

    } else {

      double price = double.parse(\_price);

      if (price <= 0) {

        isValid = false;

        \_priceShowError = true;

        \_priceError = 'Debes ingresar un Precio mayor a cero';

      } else {

        \_priceShowError = false;

      }

    }

    setState(() {});

    return isValid;

  }

  \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'description': \_description,

      'price': double.parse(\_price),

    };

    Response response =

        await ApiHelper.post('/api/Procedures/', request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context);

  }

  \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'id': widget.procedure.id,

      'description': \_description,

      'price': double.parse(\_price),

    };

    Response response = await ApiHelper.put('/api/Procedures/',

        widget.procedure.id.toString(), request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context);

  }

  void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.delete(

        '/api/Procedures/', widget.procedure.id.toString(), widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context);

  }

}

# Buscador de procedimientos

En **procedures\_screen.dart** hacemos:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/procedure\_screen.dart';

class ProceduresScreen extends StatefulWidget {

  final Token token;

  ProceduresScreen({required this.token});

  @override

  \_ProceduresScreenState createState() => \_ProceduresScreenState();

}

class \_ProceduresScreenState extends State<ProceduresScreen> {

  List<Procedure> \_procedures = [];

  bool \_showLoader = false;

  bool \_isFiltered = false;

  String \_search = '';

  @override

  void initState() {

    super.initState();

    \_getProcedures();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Procedimientos'),

        actions: <Widget>[

          \_isFiltered

              ? IconButton(

                  onPressed: \_removeFilter, icon: Icon(Icons.filter\_none))

              : IconButton(onPressed: \_showFilter, icon: Icon(Icons.filter\_alt))

        ],

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(text: 'Por favor espere...')

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () {

          Navigator.push(

              context,

              MaterialPageRoute(

                  builder: (context) => ProcedureScreen(

                      token: widget.token,

                      procedure: Procedure(description: '', id: 0, price: 0))));

        },

      ),

    );

  }

  void \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.getProcedures(widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_procedures = response.result;

    });

  }

  Widget \_getContent() {

    return \_procedures.length == 0 ? \_noContent() : \_getListView();

  }

  Widget \_noContent() {

    return Container(

      margin: EdgeInsets.all(20),

      child: Center(

        child: Text(

          \_isFiltered

              ? 'No hay procedimientos con ese criterio de búsqueda'

              : 'No hay procedimientos registrados',

          style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getListView() {

    return ListView(

      children: \_procedures.map((e) {

        return Card(

          child: InkWell(

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) =>

                          ProcedureScreen(token: widget.token, procedure: e)));

            },

            child: Container(

              margin: EdgeInsets.all(10),

              padding: EdgeInsets.all(5),

              child: Column(

                children: [

                  Row(

                    mainAxisAlignment: MainAxisAlignment.spaceBetween,

                    children: [

                      Text(

                        e.description,

                      ),

                      Icon(Icons.arrow\_forward\_ios),

                    ],

                  ),

                  SizedBox(

                    height: 15,

                  ),

                  Row(

                    children: [

                      Text(

                        '${NumberFormat.currency(symbol: '\$').format(e.price)}',

                        style: TextStyle(fontWeight: FontWeight.bold),

                      ),

                    ],

                  ),

                ],

              ),

            ),

          ),

        );

      }).toList(),

    );

  }

  void \_showFilter() {

    showDialog(

        context: context,

        builder: (context) {

          return AlertDialog(

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(10),

            ),

            title: Text('Filtrar Procedimientos'),

            content: Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Text('Escriba las primeras letras del Procedimiento'),

              SizedBox(

                height: 10,

              ),

              TextField(

                autofocus: true,

                decoration: InputDecoration(

                    hintText: 'Criterio de búsqueda...',

                    labelText: 'Buscar',

                    suffixIcon: Icon(Icons.search),

                    border: OutlineInputBorder(

                        borderRadius: BorderRadius.circular(10))),

                onChanged: (value) {

                  \_search = value;

                },

              ),

            ]),

            actions: <Widget>[

              TextButton(

                  onPressed: () => Navigator.of(context).pop(),

                  child: Text('Cancelar')),

              TextButton(onPressed: () => \_filter(), child: Text('Filtrar')),

            ],

          );

        });

  }

  void \_removeFilter() {

    setState(() {

      \_isFiltered = false;

    });

    \_getProcedures();

  }

  \_filter() {

    if (\_search.isEmpty) {

      return;

    }

    List<Procedure> filteredList = [];

    for (var procedure in \_procedures) {

      if (procedure.description.toLowerCase().contains(\_search.toLowerCase())) {

        filteredList.add(procedure);

      }

    }

    setState(() {

      \_procedures = filteredList;

      \_isFiltered = true;

    });

    Navigator.of(context).pop();

  }

}

En **procedure\_screen.dart** hacemos en los métodos **\_addRecord**, **\_editRecord** y **\_deleteRecord**::

  \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'description': \_description,

      'price': double.parse(\_price),

    };

    Response response =

        await ApiHelper.post('/api/Procedures/', request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'id': widget.procedure.id,

      'description': \_description,

      'price': double.parse(\_price),

    };

    Response response = await ApiHelper.put('/api/Procedures/',

        widget.procedure.id.toString(), request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.delete(

        '/api/Procedures/', widget.procedure.id.toString(), widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

}

En **procedures\_screen.dart** hacemos:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/procedure\_screen.dart';

class ProceduresScreen extends StatefulWidget {

  final Token token;

  ProceduresScreen({required this.token});

  @override

  \_ProceduresScreenState createState() => \_ProceduresScreenState();

}

class \_ProceduresScreenState extends State<ProceduresScreen> {

  List<Procedure> \_procedures = [];

  bool \_showLoader = false;

  bool \_isFiltered = false;

  String \_search = '';

  @override

  void initState() {

    super.initState();

    \_getProcedures();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Procedimientos'),

        actions: <Widget>[

          \_isFiltered

              ? IconButton(

                  onPressed: \_removeFilter, icon: Icon(Icons.filter\_none))

              : IconButton(onPressed: \_showFilter, icon: Icon(Icons.filter\_alt))

        ],

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(text: 'Por favor espere...')

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAdd(),

      ),

    );

  }

  Future<Null> \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.getProcedures(widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_procedures = response.result;

    });

  }

  Widget \_getContent() {

    return \_procedures.length == 0 ? \_noContent() : \_getListView();

  }

  Widget \_noContent() {

    return Container(

      margin: EdgeInsets.all(20),

      child: Center(

        child: Text(

          \_isFiltered

              ? 'No hay procedimientos con ese criterio de búsqueda'

              : 'No hay procedimientos registrados',

          style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getListView() {

    return RefreshIndicator(

      onRefresh: \_getProcedures,

      child: ListView(

        children: \_procedures.map((e) {

          return Card(

            child: InkWell(

              onTap: () => \_goEdit(e),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Column(

                  children: [

                    Row(

                      mainAxisAlignment: MainAxisAlignment.spaceBetween,

                      children: [

                        Text(

                          e.description,

                        ),

                        Icon(Icons.arrow\_forward\_ios),

                      ],

                    ),

                    SizedBox(

                      height: 15,

                    ),

                    Row(

                      children: [

                        Text(

                          '${NumberFormat.currency(symbol: '\$').format(e.price)}',

                          style: TextStyle(fontWeight: FontWeight.bold),

                        ),

                      ],

                    ),

                  ],

                ),

              ),

            ),

          );

        }).toList(),

      ),

    );

  }

  void \_showFilter() {

    showDialog(

        context: context,

        builder: (context) {

          return AlertDialog(

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(10),

            ),

            title: Text('Filtrar Procedimientos'),

            content: Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Text('Escriba las primeras letras del Procedimiento'),

              SizedBox(

                height: 10,

              ),

              TextField(

                autofocus: true,

                decoration: InputDecoration(

                    hintText: 'Criterio de búsqueda...',

                    labelText: 'Buscar',

                    suffixIcon: Icon(Icons.search),

                    border: OutlineInputBorder(

                        borderRadius: BorderRadius.circular(10))),

                onChanged: (value) {

                  \_search = value;

                },

              ),

            ]),

            actions: <Widget>[

              TextButton(

                  onPressed: () => Navigator.of(context).pop(),

                  child: Text('Cancelar')),

              TextButton(onPressed: () => \_filter(), child: Text('Filtrar')),

            ],

          );

        });

  }

  void \_removeFilter() {

    setState(() {

      \_isFiltered = false;

    });

    \_getProcedures();

  }

  \_filter() {

    if (\_search.isEmpty) {

      return;

    }

    List<Procedure> filteredList = [];

    for (var procedure in \_procedures) {

      if (procedure.description.toLowerCase().contains(\_search.toLowerCase())) {

        filteredList.add(procedure);

      }

    }

    setState(() {

      \_procedures = filteredList;

      \_isFiltered = true;

    });

    Navigator.of(context).pop();

  }

  void \_goAdd() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => ProcedureScreen(

                token: widget.token,

                procedure: Procedure(description: '', id: 0, price: 0))));

    if (result == 'yes') {

      \_getProcedures();

    }

  }

  void \_goEdit(Procedure procedure) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) =>

                ProcedureScreen(token: widget.token, procedure: procedure)));

    if (result == 'yes') {

      \_getProcedures();

    }

  }

}

# Nuevos controladores API

En el proyecto de Visual Studio Vehicles.Api creamos los API Controller BrandsController, DocumentTypesController y VehicleTypesController

Sobre la carpeta API hacemos Clic derecho – Agregar – Controlador – Api – Controlador de Api con acciones que usan EntityFramework

Son similares al ProcedureController

Las correcciones que debemos hacer son:

* Agregar la línea de autorización
* Agregar el OrderBy al método Get
* Corregir los catch de los métodos Put y Post
* Borrar el método Exists

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

public class BrandsController : ControllerBase

{

private readonly DataContext \_context;

public BrandsController(DataContext context)

{

\_context = context;

}

// GET: api/Brands

[HttpGet]

public async Task<ActionResult<IEnumerable<Brand>>> GetBrands()

{

return await \_context.Brands.OrderBy(x => x.Description).ToListAsync();

}

// GET: api/Brands/5

[HttpGet("{id}")]

public async Task<ActionResult<Brand>> GetBrand(int id)

{

Brand brand = await \_context.Brands.FindAsync(id);

if (brand == null)

{

return NotFound();

}

return brand;

}

// PUT: api/Brands/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutBrand(int id, Brand brand)

{

if (id != brand.Id)

{

return BadRequest();

}

\_context.Entry(brand).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

return NoContent();

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe esta marca.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// POST: api/Brands

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Brand>> PostBrand(Brand brand)

{

\_context.Brands.Add(brand);

try

{

await \_context.SaveChangesAsync();

return CreatedAtAction("GetBrand", new { id = brand.Id }, brand);

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe esta marca.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// DELETE: api/Brands/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteBrand(int id)

{

Brand brand = await \_context.Brands.FindAsync(id);

if (brand == null)

{

return NotFound();

}

\_context.Brands.Remove(brand);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

public class DocumentTypesController : ControllerBase

{

private readonly DataContext \_context;

public DocumentTypesController(DataContext context)

{

\_context = context;

}

// GET: api/DocumentTypes

[HttpGet]

public async Task<ActionResult<IEnumerable<DocumentType>>> GetDocumentTypes()

{

return await \_context.DocumentTypes.OrderBy(x => x.Description).ToListAsync();

}

// GET: api/DocumentTypes/5

[HttpGet("{id}")]

public async Task<ActionResult<DocumentType>> GetDocumentType(int id)

{

DocumentType documentType = await \_context.DocumentTypes.FindAsync(id);

if (documentType == null)

{

return NotFound();

}

return documentType;

}

// PUT: api/DocumentTypes/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutDocumentType(int id, DocumentType documentType)

{

if (id != documentType.Id)

{

return BadRequest();

}

\_context.Entry(documentType).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

return NoContent();

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe este tipo de documento.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// POST: api/DocumentTypes

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<DocumentType>> PostDocumentType(DocumentType documentType)

{

\_context.DocumentTypes.Add(documentType);

try

{

await \_context.SaveChangesAsync();

return CreatedAtAction("GetDocumentType", new { id = documentType.Id }, documentType);

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe esta tipo de documento.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// DELETE: api/DocumentTypes/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteDocumentType(int id)

{

DocumentType documentType = await \_context.DocumentTypes.FindAsync(id);

if (documentType == null)

{

return NotFound();

}

\_context.DocumentTypes.Remove(documentType);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Controllers.API

{

[Route("api/[controller]")]

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

public class VehicleTypesController : ControllerBase

{

private readonly DataContext \_context;

public VehicleTypesController(DataContext context)

{

\_context = context;

}

// GET: api/VehicleTypes

[HttpGet]

public async Task<ActionResult<IEnumerable<VehicleType>>> GetVehicleTypes()

{

return await \_context.VehicleTypes.OrderBy(x => x.Description).ToListAsync();

}

// GET: api/VehicleTypes/5

[HttpGet("{id}")]

public async Task<ActionResult<VehicleType>> GetVehicleType(int id)

{

VehicleType vehicleType = await \_context.VehicleTypes.FindAsync(id);

if (vehicleType == null)

{

return NotFound();

}

return vehicleType;

}

// PUT: api/VehicleTypes/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutVehicleType(int id, VehicleType vehicleType)

{

if (id != vehicleType.Id)

{

return BadRequest();

}

\_context.Entry(vehicleType).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

return NoContent();

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe este tipo de vehículo.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// POST: api/VehicleTypes

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<VehicleType>> PostVehicleType(VehicleType vehicleType)

{

\_context.VehicleTypes.Add(vehicleType);

try

{

await \_context.SaveChangesAsync();

return CreatedAtAction("GetVehicleType", new { id = vehicleType.Id }, vehicleType);

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe esta tipo de vehículo.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

// DELETE: api/VehicleTypes/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteVehicleType(int id)

{

VehicleType vehicleType = await \_context.VehicleTypes.FindAsync(id);

if (vehicleType == null)

{

return NotFound();

}

\_context.VehicleTypes.Remove(vehicleType);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

Publicamos el Api de nuevo.

# Pantallas y CRUD’s de Marcas, Tipos de Documento y Tipos de Vehículo

A través de la página JSON2Dart obtenemos las clases para crear los archivos **brand.dart,** y **vehicleType.dart**

class Brand {

  int id = 0;

  String description = '';

  Brand({required this.id, required this.description});

  Brand.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    description = json['description'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['description'] = this.description;

    return data;

  }

}

class VehicleType {

  int id = 0;

  String description = '';

  VehicleType({required this.id, required this.description});

  VehicleType.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    description = json['description'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['description'] = this.description;

    return data;

  }

}

Agregamos los métodos **getBrands, getDocumentTypes y getVehicleTypes** a **api\_helper.dart**

static Future<Response> getBrands(String token) async {

    var url = Uri.parse('${Constants.apiUrl}/api/Brands');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    List<Brand> list = [];

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        list.add(Brand.fromJson(item));

      }

    }

    return Response(isSuccess: true, result: list);

  }

  static Future<Response> getDocumentTypes(String token) async {

    var url = Uri.parse('${Constants.apiUrl}/api/DocumentTypes');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    List<DocumentType> list = [];

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        list.add(DocumentType.fromJson(item));

      }

    }

    return Response(isSuccess: true, result: list);

  }

  static Future<Response> getVehicleTypes(String token) async {

    var url = Uri.parse('${Constants.apiUrl}/api/VehicleTypes');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    List<VehicleType> list = [];

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        list.add(VehicleType.fromJson(item));

      }

    }

    return Response(isSuccess: true, result: list);

  }

Creamos los archivos en la carpeta screens:

**brand\_screen.dart**

**brands\_screen.dart**

**document\_type\_screen.dart**

**documents\_type\_screen.dart**

**vehicle\_type\_screen.dart**

**vehicles\_type\_screen.dart**

y modificamos **home\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/brands\_screen.dart';

import 'package:vehicles\_app/screens/document\_types\_screen.dart';

import 'package:vehicles\_app/screens/login\_screen.dart';

import 'package:vehicles\_app/screens/procedures\_screen.dart';

import 'package:vehicles\_app/screens/vehicle\_types\_screen.dart';

class HomeScreen extends StatefulWidget {

  final Token token;

  HomeScreen({required this.token});

  @override

  \_HomeScreenState createState() => \_HomeScreenState();

}

class \_HomeScreenState extends State<HomeScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Vehicles'),

      ),

      body: \_getBody(),

      drawer: widget.token.user2.userType == 0

          ? \_getMechanicMenu()

          : \_getCustomerMenu(),

    );

  }

  Widget \_getBody() {

    return Container(

      margin: EdgeInsets.all(30),

      child: Column(

        mainAxisAlignment: MainAxisAlignment.center,

        children: [

          ClipRRect(

            borderRadius: BorderRadius.circular(150),

            child: FadeInImage(

                placeholder: AssetImage('assets/logo.png'),

                image: NetworkImage(widget.token.user2.imageFullPath),

                height: 300,

                fit: BoxFit.cover),

          ),

          SizedBox(

            height: 30,

          ),

          Center(

            child: Text(

              'Bienvenido/a ${widget.token.user2.fullName}',

              style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

            ),

          )

        ],

      ),

    );

  }

  Widget \_getMechanicMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Marcas'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => BrandsScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.precision\_manufacturing),

            title: Text('Procedimientos'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => ProceduresScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.badge),

            title: Text('Tipos de Documento'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => DocumentTypesScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.toys),

            title: Text('Tipos de Vehículo'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => VehicleTypesScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.people),

            title: Text('Usuarios'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

            },

          ),

        ],

      ),

    );

  }

  \_getCustomerMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Mis Vehículos'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

            },

          ),

        ],

      ),

    );

  }

}

# Users Controller

En el Proyecto de Visual Studio dentro de la carpeta **Controllers/API** creamos el archivo **UsersController**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.Common.Enums;

namespace Vehicles.Api.Controllers.API

{

[ApiController]

//[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

public class UsersController : ControllerBase

{

private readonly DataContext \_context;

public UsersController(DataContext context)

{

this.\_context = context;

}

// GET: api/Users

[HttpGet]

public async Task<ActionResult<IEnumerable<User>>> GetUsers()

{

return await \_context.Users

.Include(x=>x.DocumentType)

.OrderBy(x => x.LastName)

.ThenBy(x=>x.FirstName)

.Where(x=> x.UserType==UserType.User)

.ToListAsync();

}

// GET: api/Users/5

[HttpGet("{id}")]

public async Task<ActionResult<User>> GetUser(string id)

{

User user = await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehicleType)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Brand)

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehiclePhotos)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Histories)

.ThenInclude(x => x.Details)

.ThenInclude(x => x.Procedure)

.FirstOrDefaultAsync(x => x.Id == id);

if (user == null)

{

return NotFound();

}

return user;

}

}

}

Para evitar las referencias cíclicas agregamos…

En **Brand.cs**:

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class Brand

{

public int Id { get; set; }

[Display(Name = "Marca")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Description { get; set; }

[JsonIgnore]

public ICollection<Vehicle> Vehicles { get; set; }

}

}

En **History.cs**:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class History

{

public int Id { get; set; }

[JsonIgnore]

[Display(Name = "Vehículo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Vehicle Vehicle { get; set; }

[Display(Name = "Fecha")]

[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

public DateTime Date { get; set; }

[Display(Name = "Fecha")]

[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

public DateTime DateLocal => Date.ToLocalTime();

[Display(Name = "Kilometraje")]

[DisplayFormat(DataFormatString = "{0:N0}")]

public int Mileage { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

[JsonIgnore]

[Display(Name = "Mecánico")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public User User { get; set; }

public ICollection<Detail> Details { get; set; }

[Display(Name = "N° Detalles")]

public int DetailsCount => Details == null ? 0 : Details.Count;

[Display(Name = "Total Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal TotalLabor => Details == null ? 0 : Details.Sum(x => x.LaborPrice);

[Display(Name = "Total Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal TotalSpareParts => Details == null ? 0 : Details.Sum(x => x.SparePartsPrice);

[Display(Name = "Total")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public decimal Total => Details == null ? 0 : Details.Sum(x => x.TotalPrice);

}

}

En **VehiclePhoto.cs**:

using System;

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class VehiclePhoto

{

public int Id { get; set; }

[JsonIgnore]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Vehicle Vehicle { get; set; }

[Display(Name = "Foto")]

public string ImageId { get; set; }

//TODO: Corregir ruta

[Display(Name = "Foto")]

public string ImageFullPath => string.IsNullOrEmpty(ImageId)

? $"http://keypress.serveftp.net:88/VehiclesApi/images/noimage.png"

: $"http://keypress.serveftp.net:88/VehiclesApi{ImageId.Substring(1)}";

//: $"http://keypress.serveftp.net:88/VehiclesApi/Images/vehicles/{ImageId}";

}

}

En **VehicleType.cs**:

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class VehicleType

{

public int Id { get; set; }

[Display(Name = "Tipo de vehículo")]

[MaxLength(50, ErrorMessage ="El campo {0} no puede tener más de {1} caracteres")]

[Required(ErrorMessage = "El campo {0} es obligatorio")]

public string Description { get; set; }

[JsonIgnore]

public ICollection<Vehicle> Vehicles { get; set; }

}

}

Publicamos el API

# Creación de los modelos que faltan

Hacemos un GetUser para un id y copiamos la respuesta y la pegamos en la página JSON2Dart.

Así obtenemos las clases que nos faltan.

Creamos el archivo **detail.dart:**

import 'package:vehicles\_app/models/procedure.dart';

class Detail {

  int id=0;

  Procedure procedure=Procedure(description: '',id:0,price:0);

  double laborPrice=0;

  double sparePartsPrice=0;

  double totalPrice=0;

  String remarks='';

  Detail(

      {required this.id,

      required this.procedure,

      required this.laborPrice,

      required this.sparePartsPrice,

      required this.totalPrice,

      required this.remarks});

  Detail.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    procedure = Procedure.fromJson(json['procedure']);

    laborPrice = json['laborPrice'];

    sparePartsPrice = json['sparePartsPrice'];

    totalPrice = json['totalPrice'];

    remarks = json['remarks'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['procedure'] = this.procedure.toJson();

    data['laborPrice'] = this.laborPrice;

    data['sparePartsPrice'] = this.sparePartsPrice;

    data['totalPrice'] = this.totalPrice;

    data['remarks'] = this.remarks;

    return data;

  }

}

Creamos el archivo **history.dart:**

import 'package:vehicles\_app/models/detail.dart';

class History {

  int id=0;

  String date='';

  String dateLocal='';

  int mileage=0;

  String remarks='';

  List<Detail> details=[];

  int detailsCount=0;

  double totalLabor=0;

  double totalSpareParts=0;

  double total=0;

  History(

      {required this.id,

      required this.date,

      required this.dateLocal,

      required this.mileage,

      required this.remarks,

      required this.details,

      required this.detailsCount,

      required this.totalLabor,

      required this.totalSpareParts,

      required this.total});

  History.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    date = json['date'];

    dateLocal = json['dateLocal'];

    mileage = json['mileage'];

    remarks = json['remarks'];

    if (json['details'] != null) {

      details = [];

      json['details'].forEach((v) {

        details.add(new Detail.fromJson(v));

      });

    }

    detailsCount = json['detailsCount'];

    totalLabor = json['totalLabor'];

    totalSpareParts = json['totalSpareParts'];

    total = json['total'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['date'] = this.date;

    data['dateLocal'] = this.dateLocal;

    data['mileage'] = this.mileage;

    data['remarks'] = this.remarks;

    data['details'] = this.details.map((v) => v.toJson()).toList();

    data['detailsCount'] = this.detailsCount;

    data['totalLabor'] = this.totalLabor;

    data['totalSpareParts'] = this.totalSpareParts;

    data['total'] = this.total;

    return data;

  }

}

Creamos el archivo **vehicle\_photo.dart:**

class VehiclePhoto {

  int id=0;

  String imageId='';

  String imageFullPath='';

  VehiclePhoto({required this.id,required this.imageId,required this.imageFullPath});

  VehiclePhoto.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    imageId = json['imageId'];

    imageFullPath = json['imageFullPath'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['imageId'] = this.imageId;

    data['imageFullPath'] = this.imageFullPath;

    return data;

  }

}

Creamos el archivo **vehicle.dart:**

import 'package:vehicles\_app/models/brand.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/vehicle\_photo.dart';

import 'package:vehicles\_app/models/vehicle\_type.dart';

class Vehicle {

  int id = 0;

  VehicleType vehicleType = VehicleType(id: 0, description: '');

  Brand brand = Brand(id: 0, description: '');

  int model = 0;

  String plaque = '';

  String line = '';

  String color = '';

  String remarks = '';

  List<VehiclePhoto> vehiclePhotos = [];

  int vehiclePhotosCount = 0;

  String imageFullPath = '';

  List<History> histories = [];

  int historiesCount = 0;

  Vehicle(

      {required this.id,

      required this.vehicleType,

      required this.brand,

      required this.model,

      required this.plaque,

      required this.line,

      required this.color,

      required this.remarks,

      required this.vehiclePhotos,

      required this.vehiclePhotosCount,

      required this.imageFullPath,

      required this.histories,

      required this.historiesCount});

  Vehicle.fromJson(Map<String, dynamic> json) {

    id = json['id'];

    vehicleType = new VehicleType.fromJson(json['vehicleType']);

    brand = new Brand.fromJson(json['brand']);

    model = json['model'];

    plaque = json['plaque'];

    line = json['line'];

    color = json['color'];

    remarks = json['remarks'];

    if (json['vehiclePhotos'] != null) {

      vehiclePhotos = [];

      json['vehiclePhotos'].forEach((v) {

        vehiclePhotos.add(new VehiclePhoto.fromJson(v));

      });

    }

    vehiclePhotosCount = json['vehiclePhotosCount'];

    imageFullPath = json['imageFullPath'];

    if (json['histories'] != null) {

      histories = [];

      json['histories'].forEach((v) {

        histories.add(new History.fromJson(v));

      });

    }

    historiesCount = json['historiesCount'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['id'] = this.id;

    data['vehicleType'] = this.vehicleType.toJson();

    data['brand'] = this.brand.toJson();

    data['model'] = this.model;

    data['plaque'] = this.plaque;

    data['line'] = this.line;

    data['color'] = this.color;

    data['remarks'] = this.remarks;

    data['vehiclePhotos'] = this.vehiclePhotos.map((v) => v.toJson()).toList();

    data['vehiclePhotosCount'] = this.vehiclePhotosCount;

    data['imageFullPath'] = this.imageFullPath;

    data['histories'] = this.histories.map((v) => v.toJson()).toList();

    data['historiesCount'] = this.historiesCount;

    return data;

  }

}

Creamos el archivo **user.dart:**

import 'package:vehicles\_app/models/vehicle.dart';

import 'document\_type.dart';

class User {

  String firstName = '';

  String lastName = '';

  DocumentType documentType = DocumentType(id: 0, description: '');

  String document = '';

  String address = '';

  String imageId = '';

  String imageFullPath = '';

  int userType = 0;

  String fullName = '';

  List<Vehicle> vehicles = [];

  int vehiclesCount = 0;

  String id = '';

  String userName = '';

  String email = '';

  String phoneNumber = '';

  User({

    required this.firstName,

    required this.lastName,

    required this.documentType,

    required this.document,

    required this.address,

    required this.imageId,

    required this.imageFullPath,

    required this.userType,

    required this.fullName,

    required this.vehicles,

    required this.vehiclesCount,

    required this.id,

    required this.userName,

    required this.email,

    required this.phoneNumber,

  });

  User.fromJson(Map<String, dynamic> json) {

    firstName = json['firstName'];

    lastName = json['lastName'];

    documentType = DocumentType.fromJson(json['documentType']);

    document = json['document'];

    address = json['address'];

    imageId = json['imageId'];

    imageFullPath = json['imageFullPath'];

    userType = json['userType'];

    fullName = json['fullName'];

    if (json['vehicles'] != null) {

      vehicles = [];

      json['vehicles'].forEach((v) {

        vehicles.add(new Vehicle.fromJson(v));

      });

    }

    vehiclesCount = json['vehiclesCount'];

    id = json['id'];

    userName = json['userName'];

    email = json['email'];

    phoneNumber = json['phoneNumber'];

  }

  Map<String, dynamic> toJson() {

    final Map<String, dynamic> data = new Map<String, dynamic>();

    data['firstName'] = this.firstName;

    data['lastName'] = this.lastName;

    if (this.documentType != null) {

      data['documentType'] = this.documentType.toJson();

    }

    data['document'] = this.document;

    data['address'] = this.address;

    data['imageId'] = this.imageId;

    data['imageFullPath'] = this.imageFullPath;

    data['userType'] = this.userType;

    data['fullName'] = this.fullName;

    data['vehicles'] = this.vehicles.map((v) => v.toJson()).toList();

    data['vehiclesCount'] = this.vehiclesCount;

    data['id'] = this.id;

    data['userName'] = this.userName;

    data['email'] = this.email;

    data['phoneNumber'] = this.phoneNumber;

    return data;

  }

}

# Pantalla de Usuarios

En **api\_helper.dart** agregamos el método **getUsers**

static Future<Response> getUsers(String token) async {

    var url = Uri.parse('${Constants.apiUrl}/api/Users');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer $token',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    List<User> list = [];

    var decodedJson = jsonDecode(body);

    if (decodedJson != null) {

      for (var item in decodedJson) {

        list.add(User.fromJson(item));

      }

    }

    return Response(isSuccess: true, result: list);

  }

En la carpeta **screens** creamos el archivo **user\_screen.dart**

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

class UserScreen extends StatefulWidget {

  final Token token;

  final User user;

  UserScreen({required this.token, required this.user});

  @override

  \_UserScreenState createState() => \_UserScreenState();

}

class \_UserScreenState extends State<UserScreen> {

  bool \_showLoader = false;

  String \_firstName = '';

  String \_firstNameError = '';

  bool \_firstNameShowError = false;

  TextEditingController \_firstNameController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_firstName = widget.user.firstName;

    \_firstNameController.text = \_firstName;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(

              widget.user.id.isEmpty ? 'Nuevo Usuario' : widget.user.fullName),

        ),

        body: Stack(

          children: [

            Column(

              children: <Widget>[

                \_showFirstName(),

                \_showButtons(),

              ],

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showFirstName() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        controller: \_firstNameController,

        decoration: InputDecoration(

            hintText: 'Ingresa nombres...',

            labelText: 'Nombres',

            errorText: \_firstNameShowError ? \_firstNameError : null,

            suffixIcon: Icon(Icons.person),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_firstName = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.user.id.isEmpty

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.user.id.isEmpty

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

  void \_save() {

    if (!validateFields()) {

      return;

    }

    widget.user.id.isEmpty ? \_addRecord() : \_saveRecord();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_firstName.isEmpty) {

      isValid = false;

      \_firstNameShowError = true;

      \_firstNameError = 'Debes ingresar un nombre';

    } else {

      \_firstNameShowError = false;

    }

    setState(() {});

    return isValid;

  }

  \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'firstName': \_firstName,

    };

    Response response =

        await ApiHelper.post('/api/Users/', request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'id': widget.user.id,

      'firstName': \_firstName,

    };

    Response response = await ApiHelper.put('/api/Users/',

        widget.user.id, request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.delete(

        '/api/Users/', widget.user.id, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

}

En la carpeta **screens** creamos el archivo **users\_screen.dart**

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/document\_type.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/user\_screen.dart';

class UsersScreen extends StatefulWidget {

  final Token token;

  UsersScreen({required this.token});

  @override

  \_UsersScreenState createState() => \_UsersScreenState();

}

class \_UsersScreenState extends State<UsersScreen> {

  List<User> \_users = [];

  bool \_showLoader = false;

  bool \_isFiltered = false;

  String \_search = '';

  @override

  void initState() {

    super.initState();

    \_getUsers();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Usuarios'),

        actions: <Widget>[

          \_isFiltered

              ? IconButton(

                  onPressed: \_removeFilter, icon: Icon(Icons.filter\_none))

              : IconButton(onPressed: \_showFilter, icon: Icon(Icons.filter\_alt))

        ],

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(text: 'Por favor espere...')

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAdd(),

      ),

    );

  }

  Future<Null> \_getUsers() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.getUsers(widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_users = response.result;

    });

  }

  Widget \_getContent() {

    return \_users.length == 0 ? \_noContent() : \_getListView();

  }

  Widget \_noContent() {

    return Container(

      margin: EdgeInsets.all(20),

      child: Center(

        child: Text(

          \_isFiltered

              ? 'No hay usuarios con ese criterio de búsqueda'

              : 'No hay usuarios registrados',

          style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getListView() {

    return RefreshIndicator(

      onRefresh: \_getUsers,

      child: ListView(

        children: \_users.map((e) {

          return Card(

            child: InkWell(

              onTap: () => \_goEdit(e),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Column(

                  children: [

                    Row(

                      mainAxisAlignment: MainAxisAlignment.spaceBetween,

                      children: [

                        Text(

                          e.fullName,

                        ),

                        Icon(Icons.arrow\_forward\_ios),

                      ],

                    ),

                    SizedBox(

                      height: 15,

                    ),

                    Row(

                      children: [],

                    ),

                  ],

                ),

              ),

            ),

          );

        }).toList(),

      ),

    );

  }

  void \_showFilter() {

    showDialog(

        context: context,

        builder: (context) {

          return AlertDialog(

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(10),

            ),

            title: Text('Filtrar Marcas'),

            content: Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Text(

                  'Escriba las primeras letras del Nombre o Apellido del Usuario'),

              SizedBox(

                height: 10,

              ),

              TextField(

                autofocus: true,

                decoration: InputDecoration(

                    hintText: 'Criterio de búsqueda...',

                    labelText: 'Buscar',

                    suffixIcon: Icon(Icons.search),

                    border: OutlineInputBorder(

                        borderRadius: BorderRadius.circular(10))),

                onChanged: (value) {

                  \_search = value;

                },

              ),

            ]),

            actions: <Widget>[

              TextButton(

                  onPressed: () => Navigator.of(context).pop(),

                  child: Text('Cancelar')),

              TextButton(onPressed: () => \_filter(), child: Text('Filtrar')),

            ],

          );

        });

  }

  void \_removeFilter() {

    setState(() {

      \_isFiltered = false;

    });

    \_getUsers();

  }

  \_filter() {

    if (\_search.isEmpty) {

      return;

    }

    List<User> filteredList = [];

    for (var user in \_users) {

      if (user.fullName.toLowerCase().contains(\_search.toLowerCase())) {

        filteredList.add(user);

      }

    }

    setState(() {

      \_users = filteredList;

      \_isFiltered = true;

    });

    Navigator.of(context).pop();

  }

  void \_goAdd() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => UserScreen(

                token: widget.token,

                user: User(

                  firstName: '',

                  lastName: '',

                  documentType: DocumentType(id: 0, description: ''),

                  document: '',

                  address: '',

                  imageId: '',

                  imageFullPath: '',

                  userType: 1,

                  fullName: '',

                  vehicles: [],

                  vehiclesCount: 0,

                  id: '',

                  userName: '',

                  email: '',

                  phoneNumber: '',

                ))));

    if (result == 'yes') {

      \_getUsers();

    }

  }

  void \_goEdit(User user) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => UserScreen(token: widget.token, user: user)));

    if (result == 'yes') {

      \_getUsers();

    }

  }

}

En **home\_screen.dart** hacemos:

ListTile(

            leading: Icon(Icons.people),

            title: Text('Usuarios'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => UsersScreen(

                            token: widget.token,

                          )));

            },

# Agregando datos a la pantalla de usuarios

En **users\_screen.dart** hacemos:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'package:intl/intl.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/helpers/constants.dart';

import 'package:vehicles\_app/models/document\_type.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/user\_screen.dart';

class UsersScreen extends StatefulWidget {

  final Token token;

  UsersScreen({required this.token});

  @override

  \_UsersScreenState createState() => \_UsersScreenState();

}

class \_UsersScreenState extends State<UsersScreen> {

  List<User> \_users = [];

  bool \_showLoader = false;

  bool \_isFiltered = false;

  String \_search = '';

  @override

  void initState() {

    super.initState();

    \_getUsers();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Usuarios'),

        actions: <Widget>[

          \_isFiltered

              ? IconButton(

                  onPressed: \_removeFilter, icon: Icon(Icons.filter\_none))

              : IconButton(onPressed: \_showFilter, icon: Icon(Icons.filter\_alt))

        ],

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(text: 'Por favor espere...')

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAdd(),

      ),

    );

  }

  Future<Null> \_getUsers() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.getUsers(widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_users = response.result;

    });

  }

  Widget \_getContent() {

    return \_users.length == 0 ? \_noContent() : \_getListView();

  }

  Widget \_noContent() {

    return Container(

      margin: EdgeInsets.all(20),

      child: Center(

        child: Text(

          \_isFiltered

              ? 'No hay usuarios con ese criterio de búsqueda'

              : 'No hay usuarios registrados',

          style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),

        ),

      ),

    );

  }

  Widget \_getListView() {

    return RefreshIndicator(

      onRefresh: \_getUsers,

      child: ListView(

        children: \_users.map((e) {

          return Card(

            child: InkWell(

              onTap: () => \_goEdit(e),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Row(

                  children: [

                    ClipRRect(

                      borderRadius: BorderRadius.circular(40),

                      child: FadeInImage(

                        placeholder: AssetImage('assets/logo.png'),

                        image: NetworkImage(e.imageFullPath),

                        width: 80,

                        height: 80,

                        fit: BoxFit.cover,

                      ),

                    ),

                    Expanded(

                      child: Container(

                        margin: EdgeInsets.symmetric(horizontal: 10),

                        child: Row(

                          mainAxisAlignment: MainAxisAlignment.start,

                          children: [

                            Column(

                              children: [

                                Text(e.fullName,

                                    style: TextStyle(

                                      fontSize: 20,

                                      fontWeight: FontWeight.bold,

                                    )),

                                SizedBox(

                                  height: 5,

                                ),

                                Text(e.email,

                                    style: TextStyle(

                                      fontSize: 14,

                                    )),

                                Text(e.phoneNumber,

                                    style: TextStyle(

                                      fontSize: 14,

                                    )),

                              ],

                            ),

                          ],

                        ),

                      ),

                    ),

                    Icon(Icons.arrow\_forward\_ios),

                  ],

                ),

              ),

            ),

          );

        }).toList(),

      ),

    );

  }

  void \_showFilter() {

    showDialog(

        context: context,

        builder: (context) {

          return AlertDialog(

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(10),

            ),

            title: Text('Filtrar Marcas'),

            content: Column(mainAxisSize: MainAxisSize.min, children: <Widget>[

              Text(

                  'Escriba las primeras letras del Nombre o Apellido del Usuario'),

              SizedBox(

                height: 10,

              ),

              TextField(

                autofocus: true,

                decoration: InputDecoration(

                    hintText: 'Criterio de búsqueda...',

                    labelText: 'Buscar',

                    suffixIcon: Icon(Icons.search),

                    border: OutlineInputBorder(

                        borderRadius: BorderRadius.circular(10))),

                onChanged: (value) {

                  \_search = value;

                },

              ),

            ]),

            actions: <Widget>[

              TextButton(

                  onPressed: () => Navigator.of(context).pop(),

                  child: Text('Cancelar')),

              TextButton(onPressed: () => \_filter(), child: Text('Filtrar')),

            ],

          );

        });

  }

  void \_removeFilter() {

    setState(() {

      \_isFiltered = false;

    });

    \_getUsers();

  }

  \_filter() {

    if (\_search.isEmpty) {

      return;

    }

    List<User> filteredList = [];

    for (var user in \_users) {

      if (user.fullName.toLowerCase().contains(\_search.toLowerCase())) {

        filteredList.add(user);

      }

    }

    setState(() {

      \_users = filteredList;

      \_isFiltered = true;

    });

    Navigator.of(context).pop();

  }

  void \_goAdd() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => UserScreen(

                token: widget.token,

                user: User(

                  firstName: '',

                  lastName: '',

                  documentType: DocumentType(id: 0, description: ''),

                  document: '',

                  address: '',

                  imageId: '',

                  imageFullPath: '',

                  userType: 1,

                  fullName: '',

                  vehicles: [],

                  vehiclesCount: 0,

                  id: '',

                  userName: '',

                  email: '',

                  phoneNumber: '',

                ))));

    if (result == 'yes') {

      \_getUsers();

    }

  }

  void \_goEdit(User user) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => UserScreen(token: widget.token, user: user)));

    if (result == 'yes') {

      \_getUsers();

    }

  }

}

# Agregando campos a la pantalla de usuario

En **user\_screen.dart** hacemos:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:email\_validator/email\_validator.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/document\_type.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

class UserScreen extends StatefulWidget {

  final Token token;

  final User user;

  UserScreen({required this.token, required this.user});

  @override

  \_UserScreenState createState() => \_UserScreenState();

}

class \_UserScreenState extends State<UserScreen> {

  bool \_showLoader = false;

  String \_firstName = '';

  String \_firstNameError = '';

  bool \_firstNameShowError = false;

  TextEditingController \_firstNameController = TextEditingController();

  String \_lastName = '';

  String \_lastNameError = '';

  bool \_lastNameShowError = false;

  TextEditingController \_lastNameController = TextEditingController();

  DocumentType \_documentType = DocumentType(id: 0, description: '');

  List<DocumentType> \_documentTypes = [];

  String \_document = '';

  String \_documentError = '';

  bool \_documentShowError = false;

  TextEditingController \_documentController = TextEditingController();

  String \_address = '';

  String \_addressError = '';

  bool \_addressShowError = false;

  TextEditingController \_addressController = TextEditingController();

  String \_email = '';

  String \_emailError = '';

  bool \_emailShowError = false;

  TextEditingController \_emailController = TextEditingController();

  String \_phoneNumber = '';

  String \_phoneNumberError = '';

  bool \_phoneNumberShowError = false;

  TextEditingController \_phoneNumberController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_getDocumentTypes();

    \_firstName = widget.user.firstName;

    \_firstNameController.text = \_firstName;

    \_lastName = widget.user.lastName;

    \_lastNameController.text = \_lastName;

    \_document = widget.user.document;

    \_documentController.text = \_document;

    \_documentType = widget.user.documentType;

    \_address = widget.user.address;

    \_addressController.text = \_address;

    \_email = widget.user.email;

    \_emailController.text = \_email;

    \_phoneNumber = widget.user.phoneNumber;

    \_phoneNumberController.text = \_phoneNumber;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(

              widget.user.id.isEmpty ? 'Nuevo Usuario' : widget.user.fullName),

        ),

        body: Stack(

          children: [

            SingleChildScrollView(

              child: Column(

                children: <Widget>[

                  \_showPhoto(),

                  \_showFirstName(),

                  \_showLastName(),

                  \_showDocumentType(),

                  \_showDocument(),

                  \_showAddress(),

                  \_showEmail(),

                  \_showPhoneNumber(),

                  \_showButtons(),

                ],

              ),

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.user.id.isEmpty

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.user.id.isEmpty

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

  void \_save() {

    if (!validateFields()) {

      return;

    }

    widget.user.id.isEmpty ? \_addRecord() : \_saveRecord();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_firstName.isEmpty) {

      isValid = false;

      \_firstNameShowError = true;

      \_firstNameError = 'Debes ingresar un nombre';

    } else {

      \_firstNameShowError = false;

    }

    if (\_lastName.isEmpty) {

      isValid = false;

      \_lastNameShowError = true;

      \_lastNameError = 'Debes ingresar un apellido';

    } else {

      \_lastNameShowError = false;

    }

    if (\_document.isEmpty) {

      isValid = false;

      \_documentShowError = true;

      \_documentError = 'Debes ingresar un documento';

    } else {

      \_documentShowError = false;

    }

    if (\_address.isEmpty) {

      isValid = false;

      \_addressShowError = true;

      \_addressError = 'Debes ingresar una dirección';

    } else {

      \_addressShowError = false;

    }

    if (\_email.isEmpty) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar tu Email';

    } else if (!EmailValidator.validate(\_email)) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar un Email válido';

    } else {

      \_emailShowError = false;

    }

    if (\_phoneNumber.isEmpty) {

      isValid = false;

      \_phoneNumberShowError = true;

      \_phoneNumberError = 'Debes ingresar un teléfono';

    } else {

      \_phoneNumberShowError = false;

    }

    setState(() {});

    return isValid;

  }

  \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'firstName': \_firstName,

      'lastName': \_lastName,

      'typeDocument': 1,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

    };

    Response response =

        await ApiHelper.post('/api/Users/', request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'id': widget.user.id,

      'firstName': \_firstName,

      'lastName': \_lastName,

      'typeDocument': 1,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

    };

    Response response = await ApiHelper.put(

        '/api/Users/', widget.user.id, request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.delete(

        '/api/Users/', widget.user.id, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  Widget \_showPhoto() {

    print(widget.user.imageFullPath);

    return Container(

      margin: EdgeInsets.only(top: 10),

      child: widget.user.id.isEmpty

          ? Image(

              image: AssetImage('assets/noimage.png'),

              width: 160,

              height: 160,

            )

          : ClipRRect(

              borderRadius: BorderRadius.circular(80),

              child: FadeInImage(

                placeholder: AssetImage('assets/logo.png'),

                image: NetworkImage(widget.user.imageFullPath),

                width: 160,

                height: 160,

                fit: BoxFit.cover,

              ),

            ),

    );

  }

  Widget \_showFirstName() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        controller: \_firstNameController,

        decoration: InputDecoration(

            hintText: 'Ingresa nombres...',

            labelText: 'Nombres',

            errorText: \_firstNameShowError ? \_firstNameError : null,

            suffixIcon: Icon(Icons.person),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_firstName = value;

        },

      ),

    );

  }

  Widget \_showLastName() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_lastNameController,

        decoration: InputDecoration(

            hintText: 'Ingresa nombres...',

            labelText: 'Apellido',

            errorText: \_lastNameShowError ? \_lastNameError : null,

            suffixIcon: Icon(Icons.person),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_lastName = value;

        },

      ),

    );

  }

  Widget \_showDocumentType() {

    return Container();

  }

  Widget \_showDocument() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_documentController,

        decoration: InputDecoration(

            hintText: 'Ingresa documento...',

            labelText: 'Documento',

            errorText: \_documentShowError ? \_documentError : null,

            suffixIcon: Icon(Icons.assignment\_ind),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_document = value;

        },

      ),

    );

  }

  Widget \_showAddress() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_addressController,

        keyboardType: TextInputType.streetAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa dirección...',

            labelText: 'Dirección',

            errorText: \_addressShowError ? \_addressError : null,

            suffixIcon: Icon(Icons.home),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_address = value;

        },

      ),

    );

  }

  Widget \_showEmail() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_emailController,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa Email...',

            labelText: 'Email',

            errorText: \_emailShowError ? \_emailError : null,

            suffixIcon: Icon(Icons.email),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_email = value;

        },

      ),

    );

  }

  Widget \_showPhoneNumber() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_phoneNumberController,

        keyboardType: TextInputType.phone,

        decoration: InputDecoration(

            hintText: 'Ingresa Teléfono...',

            labelText: 'Teléfono',

            errorText: \_phoneNumberShowError ? \_phoneNumberError : null,

            suffixIcon: Icon(Icons.phone),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_phoneNumber = value;

        },

      ),

    );

  }

  Future<Null> \_getDocumentTypes() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.getDocumentTypes(widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_documentTypes = response.result;

    });

  }

}

# Arreglo Pantalla Login

En **login\_screen.dart** hacemos:

@override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Stack(

        children: <Widget>[

          SingleChildScrollView(

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                SizedBox(

                  height: 60,

                ),

                \_showLogo(),

# Modificación Pantalla Usuario

En **user\_screen.dart** hacemos:

int \_documentTypeId = 0;

  String \_documentTypeIdError = '';

  bool \_documentTypeIdShowError = false;

  List<DocumentType> \_documentTypes = [];

  \_documentTypeId = widget.user.documentType.id;

bool validateFields() {

    bool isValid = true;

    ……………….

    if (\_documentTypeId == 0) {

      isValid = false;

      \_documentTypeIdShowError = true;

      \_documentTypeIdError = 'Debes seleccionar un Tipo de Documento';

    } else {

      \_documentTypeIdShowError = false;

    }

\_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'firstName': \_firstName,

      'lastName': \_lastName,

      'typeDocument': \_documentTypeId,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

    };

\_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'id': widget.user.id,

      'firstName': \_firstName,

      'lastName': \_lastName,

      'typeDocument': \_documentTypeId,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

    };

  Widget \_showDocumentType() {

    return Container(

      padding: EdgeInsets.all(10),

      child: \_documentTypes.length == 0

          ? Text('Cargando tipos de documento...')

          : DropdownButtonFormField(

              items: \_getComboDocumentTypes(),

              value: \_documentTypeId,

              onChanged: (option) {

                setState(() {

                  \_documentTypeId = option as int;

                });

              },

              decoration: InputDecoration(

                hintText: 'Seleccione un tipo de documento...',

                labelText: 'Tipo de Documento',

                errorText:

                    \_documentTypeIdShowError ? \_documentTypeIdError : null,

                border:

                    OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

              )),

    );

  }

  List<DropdownMenuItem<int>> \_getComboDocumentTypes() {

    List<DropdownMenuItem<int>> list = [];

    list.add(DropdownMenuItem(

      child: Text('Seleccione un Tipo de Documento...'),

      value: 0,

    ));

    \_documentTypes.forEach((documentType) {

      list.add(DropdownMenuItem(

        child: Text(documentType.description),

        value: documentType.id,

      ));

    });

    return list;

  }

}

# Acceder a la cámara

Guía:

<https://flutter.dev/docs/cookbook/plugins/picture-using-camera>

Pegamos en el archivo **pubspec.yaml**

# The following adds the Cupertino Icons font to your application.

  # Use with the CupertinoIcons class for iOS style icons.

  cupertino\_icons: ^1.0.2

  email\_validator: ^2.0.1

  http: ^0.13.3

  intl: ^0.17.0

  adaptive\_dialog: ^1.1.0

  camera: ^0.9.4

  path\_provider: ^2.0.5

  path: ^1.8.0

En la carpeta **android/app** está el archivo **build.gradle**. Ahí dentro debemos poner:

defaultConfig {

        // TODO: Specify your own unique Application ID (https://developer.android.com/studio/build/application-id.html).

        applicationId "com.example.vehicles\_app"

        minSdkVersion 21

        targetSdkVersion 30

        versionCode flutterVersionCode.toInteger()

        versionName flutterVersionName

    }

En la carpeta **screens** creamos el archivo **display\_picture\_screen.dart**:

import 'dart:io';

import 'package:flutter/material.dart';

class DisplayPictureScreen extends StatefulWidget {

  final String imagePath;

  DisplayPictureScreen({required this.imagePath});

  @override

  \_DisplayPictureScreenState createState() => \_DisplayPictureScreenState();

}

class \_DisplayPictureScreenState extends State<DisplayPictureScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Vista previa de la foto'),

      ),

      body: Image.file(File(widget.imagePath)),

    );

  }

}

En la carpeta **screens** creamos el archivo **take\_picture.dart**

import 'package:camera/camera.dart';

import 'package:flutter/material.dart';

import 'display\_picture\_screen.dart';

class TakePictureScreen extends StatefulWidget {

  final CameraDescription camera;

  TakePictureScreen({required this.camera});

  @override

  \_TakePictureScreenState createState() => \_TakePictureScreenState();

}

class \_TakePictureScreenState extends State<TakePictureScreen> {

  late CameraController \_controller;

  late Future<void> \_initializeControllerFuture;

  @override

  void initState() {

    super.initState();

    \_controller = CameraController(

      widget.camera,

      ResolutionPreset.low,

    );

    \_initializeControllerFuture = \_controller.initialize();

  }

  @override

  void dispose() {

    \_controller.dispose();

    super.dispose();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Tomar Foto'),

      ),

      body:FutureBuilder<void> (

        future: \_initializeControllerFuture,

        builder: (context,snapshot){

          if(snapshot.connectionState == ConnectionState.done){

            return CameraPreview(\_controller);

          } else {

            return Center(

              child: CircularProgressIndicator(),

            );

          }

        },

        ),

        floatingActionButton: FloatingActionButton(

          child:Icon(Icons.camera\_alt),

          onPressed: () async{

            try {

              await \_initializeControllerFuture;

              final image=await \_controller.takePicture();

              await Navigator.of(context).push(

                MaterialPageRoute(

                  builder:(context) => DisplayPictureScreen(imagePath:image.path,)

                  )

                );

            } catch (e) {

              print(e);

            }

          },),

    );

  }

}

En **user\_screen.dart** hacemos:

  Widget \_showPhoto() {

    return InkWell(

      onTap:() => \_takePicture(),

      child: Container(

        margin: EdgeInsets.only(top: 10),

        child: widget.user.id.isEmpty

            ? Image(

                image: AssetImage('assets/noimage.png'),

                width: 160,

                height: 160,

              )

            : ClipRRect(

                borderRadius: BorderRadius.circular(80),

                child: FadeInImage(

                  placeholder: AssetImage('assets/logo.png'),

                  image: NetworkImage(widget.user.imageFullPath),

                  width: 160,

                  height: 160,

                  fit: BoxFit.cover,

                ),

              ),

      ),

    );

  }

void \_takePicture() async {

    WidgetsFlutterBinding.ensureInitialized();

    final cameras = await availableCameras();

    final firstCamera = cameras.first;

    Navigator.push(

      context,

      MaterialPageRoute(

        builder: (context) => TakePictureScreen(camera: firstCamera,)

      )

    );

  }

}

 Widget \_showPhoto() {

    return InkWell(

      onTap: () => \_takePicture(),

      child: Stack(children: <Widget>[

        Container(

          margin: EdgeInsets.only(top: 10),

          child: widget.user.id.isEmpty

              ? Image(

                  image: AssetImage('assets/noimage.png'),

                  width: 160,

                  height: 160,

                )

              : ClipRRect(

                  borderRadius: BorderRadius.circular(80),

                  child: FadeInImage(

                    placeholder: AssetImage('assets/logo.png'),

                    image: NetworkImage(widget.user.imageFullPath),

                    width: 160,

                    height: 160,

                    fit: BoxFit.cover,

                  ),

                ),

        ),

        Positioned(

            bottom: 0,

            left: 100,

            child: ClipRRect(

              borderRadius: BorderRadius.circular(30),

              child: Container(

                color: Colors.green[50],

                height: 60,

                width: 60,

                child: Icon(

                  Icons.photo\_camera,

                  size: 40,

                  color: Colors.blue,

                ),

              ),

            )),

      ]),

    );

  }

En **display\_picture\_screen.dart** hacemos:

import 'dart:io';

import 'package:camera/camera.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/response.dart';

class DisplayPictureScreen extends StatefulWidget {

  final XFile image;

  DisplayPictureScreen({required this.image});

  @override

  \_DisplayPictureScreenState createState() => \_DisplayPictureScreenState();

}

class \_DisplayPictureScreenState extends State<DisplayPictureScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Vista previa de la foto'),

      ),

      body: Column(

        children: [

          Image.file(

            File(widget.image.path),

            width: MediaQuery.of(context).size.width,

          ),

          Container(

              margin: EdgeInsets.all(10),

              child: Row(

                children: <Widget>[

                  Expanded(

                    child: ElevatedButton(

                      child: Text('Usar Foto'),

                      style: ButtonStyle(

                        backgroundColor:

                            MaterialStateProperty.resolveWith<Color>(

                                (Set<MaterialState> states) {

                          return Color(0xFF120E43);

                        }),

                      ),

                      onPressed: () {

                        Response response =

                            Response(isSuccess: true, result: widget.image);

                        Navigator.pop(context, response);

                      },

                    ),

                  ),

                  SizedBox(

                    width: 10,

                  ),

                  Expanded(

                    child: ElevatedButton(

                      child: Text('Volver a tomar'),

                      style: ButtonStyle(

                        backgroundColor:

                            MaterialStateProperty.resolveWith<Color>(

                                (Set<MaterialState> states) {

                          return Color(0xFFE03B8B);

                        }),

                      ),

                      onPressed: () {

                        Response response =

                            Response(isSuccess: true, result: widget.image);

                        Navigator.pop(context);

                      },

                    ),

                  ),

                ],

              )),

        ],

      ),

    );

  }

}

En **take\_picture.dart** modificamos:

floatingActionButton: FloatingActionButton(

        child: Icon(Icons.camera\_alt),

        onPressed: () async {

          try {

            await \_initializeControllerFuture;

            final image = await \_controller.takePicture();

            Response? response =

                await Navigator.of(context).push(MaterialPageRoute(

                    builder: (context) => DisplayPictureScreen(

                          image: image,

                        )));

            if (response != null) {

              Navigator.pop(context, response);

            }

          } catch (e) {

            print(e);

          }

        },

      ),

        },

      ),

En **user\_screen.dart** hacemos:

class \_UserScreenState extends State<UserScreen> {

  bool \_showLoader = false;

  bool \_photoChanged = false;

  late XFile \_image;

Widget \_showPhoto() {

    return InkWell(

      onTap: () => \_takePicture(),

      child: Stack(children: <Widget>[

        Container(

          margin: EdgeInsets.only(top: 10),

          child: widget.user.id.isEmpty && !\_photoChanged

              ? Image(

                  image: AssetImage('assets/noimage.png'),

                  width: 160,

                  height: 160,

fit: BoxFit.cover

                )

              : ClipRRect(

                  borderRadius: BorderRadius.circular(80),

                  child: \_photoChanged

                  ? Image.file(

                    File(\_image.path),

                    width: 160,

                    height: 160,

fit: BoxFit.cover

                  )

                  : FadeInImage(

                    placeholder: AssetImage('assets/logo.png'),

                    image: NetworkImage(widget.user.imageFullPath),

                    width: 160,

                    height: 160,

                    fit: BoxFit.cover,

                  ),

                ),

        ),

        Positioned(

            bottom: 0,

            left: 100,

            child: ClipRRect(

              borderRadius: BorderRadius.circular(30),

              child: Container(

                color: Colors.green[50],

                height: 60,

                width: 60,

                child: Icon(

                  Icons.photo\_camera,

                  size: 40,

                  color: Colors.blue,

                ),

              ),

            )),

      ]),

    );

  }

void \_takePicture() async {

    WidgetsFlutterBinding.ensureInitialized();

    final cameras = await availableCameras();

    final firstCamera = cameras.first;

    Response? response = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => TakePictureScreen(

                  camera: firstCamera,

                )));

    if (response != null) {

      setState(() {

        \_photoChanged = true;

        \_image = response.result;

      });

    }

  }

En **display\_picture\_screen.dart** hacemos:

class \_DisplayPictureScreenState extends State<DisplayPictureScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Vista previa de la foto'),

      ),

      body: Column(

        children: [

          Image.file(

            File(widget.image.path),

            width: MediaQuery.of(context).size.width,

            fit: BoxFit.cover,

# Métodos para guardar, modificar y borrar Usuario

Vamos al Proyecto de Visual Studio

En **Vehicles.Api/Models** creamos la carpeta **Request** y dentro creamos la Clase **UserRequest**:

using System.ComponentModel.DataAnnotations;

namespace Vehicles.API.Models.Request

{

public class UserRequest

{

public string Id { get; set; }

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Email { get; set; }

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string FirstName { get; set; }

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string LastName { get; set; }

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Document { get; set; }

[MaxLength(100, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Address { get; set; }

[MaxLength(20, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string PhoneNumber { get; set; }

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int DocumentTypeId { get; set; }

public byte[] Image { get; set; }

}

}

En el **UserControllers** del API agregamos el método **PostUser**:

[HttpPost]

public async Task<ActionResult<User>> PostUser(UserRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

DocumentType documentType = await \_context.DocumentTypes.FindAsync(request.DocumentTypeId);

if (documentType == null)

{

return BadRequest("El tipo de documento no existe.");

}

User user = await \_userHelper.GetUserAsync(request.Email);

if (user != null)

{

return BadRequest("Ya existe un usuario registrado con ese email.");

}

string imageId = string.Empty;

if (request.Image != null && request.Image.Length > 0)

{

imageId = \_imageHelper.UploadImage(request.Image, "users");

}

user = new User

{

Address = request.Address,

Document = request.Document,

DocumentType = documentType,

Email = request.Email,

FirstName = request.FirstName,

ImageId = imageId,

LastName = request.LastName,

PhoneNumber = request.PhoneNumber,

UserName = request.Email,

UserType = UserType.User,

};

await \_userHelper.AddUserAsync(user, "123456");

await \_userHelper.AddUserToRoleAsync(user, user.UserType.ToString());

string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

string tokenLink = Url.Action("ConfirmEmail", "Account", new

{

userid = user.Id,

token = myToken

}, protocol: HttpContext.Request.Scheme);

\_mailHelper.SendMail(request.Email, "Vehicles - Confirmación de cuenta", $"<h1>Vehicles - Confirmación de cuenta</h1>" +

$"Para habilitar el usuario, " +

$"por favor hacer clic en el siguiente enlace: </br></br><a href = \"{tokenLink}\">Confirmar Email</a>");

return Ok(user);

}

En el **UserControllers** del API agregamos el método **PutUser**:

// PUT: api/Users/5

[HttpPut("{id}")]

public async Task<IActionResult> PutUser(string id, UserRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

DocumentType documentType = await \_context.DocumentTypes.FindAsync(request.DocumentTypeId);

if (documentType == null)

{

return BadRequest("El tipo de documento no existe.");

}

User user = await \_userHelper.GetUserAsync(request.Email);

if (user == null)

{

return BadRequest("No existe el usuario.");

}

string imageId = user.ImageId;

if (request.Image != null && request.Image.Length > 0)

{

imageId = \_imageHelper.UploadImage(request.Image, "users");

}

user.Address = request.Address;

user.Document = request.Document;

user.DocumentType = documentType;

user.FirstName = request.FirstName;

user.ImageId = imageId;

user.LastName = request.LastName;

user.PhoneNumber = request.PhoneNumber;

await \_userHelper.UpdateUserAsync(user);

return NoContent();

}

En el **UserControllers** del API agregamos el método **DeleteUser**:

// DELETE: api/Users/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteUser(string id)

{

User user = await \_context.Users

.Include(x => x.Vehicles)

.ThenInclude(x => x.VehiclePhotos)

.Include(x => x.Vehicles)

.ThenInclude(x => x.Histories)

.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x=> x.Id==id);

if (user == null)

{

return NotFound();

}

\_context.Users.Remove(user);

await \_context.SaveChangesAsync();

return NoContent();

}

Publicamos el API

# Pantalla User Info

En **user\_screen.dart** hacemos:

\_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    String base64image = '';

    if (\_photoChanged) {

      List<int> imageBytes = await \_image.readAsBytes();

      base64image = base64Encode(imageBytes);

    }

    Map<String, dynamic> request = {

      'firstName': \_firstName,

      'lastName': \_lastName,

      'documentTypeId': \_documentTypeId,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

      'image':base64image,

    };

    Response response =

        await ApiHelper.post('/api/Users/', request, widget.token.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

 String base64image = '';

    if (\_photoChanged) {

      List<int> imageBytes = await \_image.readAsBytes();

      base64image = base64Encode(imageBytes);

    }

    Map<String, dynamic> request = {

      'id': widget.user.id,

      'firstName': \_firstName,

      'lastName': \_lastName,

      'documentTypeId': \_documentTypeId,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

      'image':base64image,

    };

También agregamos en **\_showEmail**:

Widget \_showEmail() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        enabled:widget.user.id.isEmpty,

        controller: \_emailController,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa Email...',

            labelText: 'Email',

            errorText: \_emailShowError ? \_emailError : null,

            suffixIcon: Icon(Icons.email),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_email = value;

        },

      ),

    );

  }

Dentro de la carpeta **screens** creamos el archivo **user\_info\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

class UserInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  UserInfoScreen({required this.token, required this.user});

  @override

  \_UserInfoScreenState createState() => \_UserInfoScreenState();

}

class \_UserInfoScreenState extends State<UserInfoScreen> {

  bool \_showLoader = false;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(widget.user.fullName),

      ),

      body: Stack(

        children: <Widget>[

          \_showUserInfo(),

          \_showButtons(),

        ],

      ),

    );

  }

  Widget \_showUserInfo() {

    return Container();

  }

  Widget \_showButtons() {

    return Container();

  }

}

En **users\_screen** modificamos **\_goEdit** por **\_goInfoUser**:

Widget \_getListView() {

    return RefreshIndicator(

      onRefresh: \_getUsers,

      child: ListView(

        children: \_users.map((e) {

          return Card(

            child: InkWell(

              onTap: () => \_goInfoUser(e),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Row(

…..

void \_goInfoUser(User user) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) =>

                UserInfoScreen(token: widget.token, user: user)));

    if (result == 'yes') {

      \_getUsers();

    }

  }

# Chequear conexión a Internet

Vamos a la página

<https://pub.dev/>

Buscamos **connectivity**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

connectivity: ^3.0.6

Copiamos la línea

connectivity: ^3.0.6

y la pegamos en el archivo **pubspec.yaml**

Agregamos estas líneas antes de todos los lugares donde se llame al apiHelper:

var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

# Seleccionar imagen desde la Galería

Vamos a la página

<https://pub.dev/>

Buscamos **image\_picker**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

image\_picker: ^0.8.4+2

Copiamos la línea

image\_picker: ^0.8.4+2

y la pegamos en el archivo **pubspec.yaml**

En **user\_screen.dart** modificamos el método **\_showPhoto**:

Widget \_showPhoto() {

    return InkWell(

      child: Stack(children: <Widget>[

        Container(

          margin: EdgeInsets.only(top: 10),

          child: widget.user.id.isEmpty && !\_photoChanged

              ? Image(

                  image: AssetImage('assets/noimage.png'),

                  width: 160,

                  height: 160,

                  fit: BoxFit.cover)

              : ClipRRect(

                  borderRadius: BorderRadius.circular(80),

                  child: \_photoChanged

                      ? Image.file(

                          File(\_image.path),

                          width: 160,

                          height: 160,

                          fit: BoxFit.cover,

                        )

                      : FadeInImage(

                          placeholder: AssetImage('assets/logo.png'),

                          image: NetworkImage(widget.user.imageFullPath),

                          width: 160,

                          height: 160,

                          fit: BoxFit.cover,

                        ),

                ),

        ),

        Positioned(

            bottom: 0,

            left: 100,

            child: InkWell(

              onTap: () => \_takePicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.photo\_camera,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

        Positioned(

            bottom: 0,

            left: 0,

            child: InkWell(

              onTap: () => \_selectPicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.image,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

      ]),

    );

  }

Y creamos el método **\_selectPicture**:

void \_selectPicture() async {

    final ImagePicker \_picker = ImagePicker();

    final XFile? image = await \_picker.pickImage(source: ImageSource.gallery);

    if (image != null) {

      setState(() {

        \_photoChanged = true;

        \_image = image;

      });

    }

  }

# Rememberme

Vamos a la página

<https://pub.dev/>

Buscamos **shared\_preferences**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

shared\_preferences: ^2.0.8

Copiamos la línea

shared\_preferences: ^2.0.8

y la pegamos en el archivo **pubspec.yaml**

En la carpeta **screens** creamos el archivo **wait\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

class WaitScreen extends StatelessWidget {

  const WaitScreen({ Key? key }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body:LoaderComponent(text:'Por favor espere...')

    );

  }

}

Modificamos en **login\_screen.dart**:

En el método **\_login** hacemos:

void \_login() async {

    setState(() {

      \_passwordShow = false;

    });

    if (!validateFields()) {

      return;

    }

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Map<String, dynamic> request = {

      'userName': \_email,

      'password': \_password,

    };

    var url = Uri.parse('${Constants.apiUrl}/api/Account/CreateToken');

    var response = await http.post(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

      },

      body: jsonEncode(request),

    );

    setState(() {

      \_showLoader = false;

    });

    if (response.statusCode >= 400) {

      setState(() {

        \_emailShowError = true;

        \_emailError = 'Email o Contraseña no válidos';

        \_passwordShowError = true;

        \_passwordError = 'Email o Contraseña no válidos';

      });

      return;

    }

    var body = response.body;

    if (\_rememberme) {

      \_storeUser(body);

    }

    var decodedJson = jsonDecode(body);

    var token = Token.fromJson(decodedJson);

    Navigator.pushReplacement(

        context,

        MaterialPageRoute(

            builder: (context) => HomeScreen(

                  token: token,

                )));

  }

Creamos el método **\_storeUser**:

void \_storeUser(String body) async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    await prefs.setBool('isRemembered',true);

    await prefs.setString('userBody',body);

  }

En **main.dart** hacemos:

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/screens/home\_screen.dart';

import 'package:vehicles\_app/screens/login\_screen.dart';

import 'package:vehicles\_app/screens/wait\_screen.dart';

void main() => runApp(MyApp());

class MyApp extends StatefulWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  \_MyAppState createState() => \_MyAppState();

}

class \_MyAppState extends State<MyApp> {

  bool \_isLoading = true;

  bool \_showLoaginPage = true;

  late Token \_token;

  @override

  void initState() {

    super.initState();

    \_getHome();

  }

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      debugShowCheckedModeBanner: false,

      title: 'Vehicles App',

      home: \_isLoading

          ? WaitScreen()

          : \_showLoaginPage

              ? LoginScreen()

              : HomeScreen(token: \_token),

    );

  }

  void \_getHome() async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    bool isRemembered = prefs.getBool('isRemembered') ?? false;

    if (isRemembered) {

      String? userBody = prefs.getString('userBody');

      if (userBody != null) {

        var decodedJson = jsonDecode(userBody);

        \_token = Token.fromJson(decodedJson);

        if (DateTime.parse(\_token.expiration).isAfter(DateTime.now())) {

          \_showLoaginPage = false;

        }

      }

    }

    \_isLoading = false;

    setState(() {});

  }

}

En **home\_screen.dart** hacemos:

Widget \_getMechanicMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Marcas'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => BrandsScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.precision\_manufacturing),

            title: Text('Procedimientos'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => ProceduresScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.badge),

            title: Text('Tipos de Documento'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => DocumentTypesScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.toys),

            title: Text('Tipos de Vehículo'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => VehicleTypesScreen(

                            token: widget.token,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.people),

            title: Text('Usuarios'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => UsersScreen(

                            token: widget.token,

                          )));

            },

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              \_logOut();

            },

          ),

        ],

      ),

    );

  }

  \_getCustomerMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Mis Vehículos'),

            onTap: () {},

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

            onTap: () {},

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              \_logOut();

            },

          ),

        ],

      ),

    );

  }

  void \_logOut() async {

 SharedPreferences prefs = await SharedPreferences.getInstance();

    await prefs.setBool('isRemembered', false);

    await prefs.setString('userBody', '');

    Navigator.pushReplacement(context,

                  MaterialPageRoute(builder: (context) => LoginScreen()));

  }

}

# Vencimiento del Token

En **api\_helper.dart** cambiamos **String token** por **Token token** (son varios lugares)

Cambiamos también **bearer** **$token** por **bearer ${token.token}**

A todos los métodos le ponemos

if (!\_validateToken(token)) {

      return Response(

          isSuccess: false,

          message:

              'Sus credenciales se han vencido, por favor cierre sesión y vuelva a ingresar al sistema.');

    }

Agregamos el método **\_validateToken**

  static bool \_validateToken(Token token) {

    if (DateTime.parse(token.expiration).isAfter(DateTime.now())) {

      return true;

    }

    return false;

  }

Cambiamos en todos lados **widget.token.token** por **widget.token** (son muchos lugares).

# Icono y Splah

En la carpeta F:\Flutter\vehicles\_app\**android\app\src\main\res** están las carpetas mimap

Dentro de cada una ponemos el archivo **logo.png** que será el splash

Luego ponemos en estos dos archivos:



En Android Asset Studio generamos los íconos, y los ponemos en las carpetas mimap:



# Traer información completa de un Usuario

En **api\_helper.dart** creamos el método **getUser**

static Future<Response> getUser(Token token,String id) async {

    if (!\_validateToken(token)) {

      return Response(

          isSuccess: false,

          message:

              'Sus credenciales se han vencido, por favor cierre sesión y vuelva a ingresar al sistema.');

    }

    var url = Uri.parse('${Constants.apiUrl}/api/Users/$id');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${token.token}',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    var decodedJson = jsonDecode(body);

    return Response(isSuccess: true, result: User.fromJson(decodedJson));

  }

En los modelos **detail.dart**, **history.dart** y **vehicle dart** ponemos:

String? remarks = '';

Ahora modificamos la **user\_info\_screen.dart**

class \_UserInfoScreenState extends State<UserInfoScreen> {

  bool \_showLoader = false;

  late User \_user;

Cambiamos

widget.user

por

\_user

en todos lados menos en

@override

  void initState() {

    super.initState();

    \_user = widget.user;

    \_getUser();

  }

Ponemos:

class \_UserInfoScreenState extends State<UserInfoScreen> {

  bool \_showLoader = false;

  late User \_user;

  @override

  void initState() {

    super.initState();

    \_user = widget.user;

    \_getUser();

  }

Y creamos el método **\_getUser**:

Future<Null> \_getUser() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.getUser(widget.token, \_user.id);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_user = response.result;

    });

  }

Ahora modificaremos la pantalla:



El botón Editar usuario lo cambiaremos por un botón de un lápiz sobre la foto, y el botón Agregar Vehículo lo cambiaremos por un botón flotante.

Widget \_showUserInfo() {

    return Container(

      margin: EdgeInsets.all(10),

      padding: EdgeInsets.all(5),

      child: Row(

        children: <Widget>[

          Stack(

            children: <Widget>[

              ClipRRect(

                borderRadius: BorderRadius.circular(50),

                child: FadeInImage(

                    placeholder: AssetImage('assets/vehicles\_logo.png'),

                    image: NetworkImage(\_user.imageFullPath),

                    width: 100,

                    height: 100,

                    fit: BoxFit.cover),

              ),

              Positioned(

                  bottom: 0,

                  left: 60,

                  child: InkWell(

                    onTap: () => \_goEdit(),

                    child: ClipRRect(

                      borderRadius: BorderRadius.circular(30),

                      child: Container(

                        color: Colors.green[50],

                        height: 40,

                        width: 40,

                        child: Icon(

                          Icons.edit,

                          size: 30,

                          color: Colors.blue,

                        ),

                      ),

                    ),

                  )),

            ],

          ),

          Expanded(

 @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(\_user.fullName),

      ),

      body: Stack(

        children: <Widget>[

          Column(

            children: <Widget>[

              \_showUserInfo(),

              \_showButtons(),

            ],

          ),

          \_showLoader

              ? LoaderComponent(

                  text: 'Por favor espere...',

                )

              : Container(),

        ],

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAdd(),),

    );

  }

Eliminamos el método \_**showButtons**, \_**showEditUserButton** y \_**showAddVehicleButton**

# Listar los vehículos de un Usuario

Corregimos en **user\_info\_screen**:

 @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(\_user.fullName),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(

                text: 'Por favor espere...',

              )

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAdd(),

      ),

    );

  }

Y creamos los métodos **\_getContent**, **\_noContent** y **\_getListView**

Widget \_getContent() {

    return Column(

      children: <Widget>[

        \_showUserInfo(),

        Expanded(

          child: \_user.vehicles.length == 0 ? \_noContent() : \_getListView(),

        )

      ],

    );

  }

  Widget \_noContent() {

    return Center(

      child: Container(

        margin: EdgeInsets.all(20),

        child: Text(

          'El usuario no tiene vehículos registrados.',

          style: TextStyle(

            fontSize: 16,

            fontWeight: FontWeight.bold,

          ),

        ),

      ),

    );

  }

  Widget \_getListView() {

    return RefreshIndicator(

        onRefresh: \_getUser,

        child: ListView(

          children: \_user.vehicles.map((e) {

            return Card(

                child: InkWell(

              onTap: () => \_goVehicle(),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Row(

                  children: <Widget>[

                    FadeInImage(

                        placeholder: AssetImage('assets/logo.png'),

                        image: NetworkImage(e.imageFullPath),

                        width: 120,

                        height: 80,

                        fit: BoxFit.cover),

                    Expanded(

                        child: Container(

                      margin: EdgeInsets.symmetric(horizontal: 10),

                      child: Row(

                        mainAxisAlignment: MainAxisAlignment.start,

                        children: <Widget>[

                          Column(

                            children: <Widget>[

                              Text(

                                e.plaque,

                                style: TextStyle(

                                    fontSize: 20, fontWeight: FontWeight.bold),

                              ),

                              Row(

                                children: [

                                  Text(

                                    e.vehicleType.description,

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                  SizedBox(

                                    width: 5,

                                  ),

                                  Text(

                                    e.brand.description,

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                ],

                              ),

                              Row(

                                children: [

                                  Text(

                                    e.line,

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                  SizedBox(

                                    width: 5,

                                  ),

                                  Text(

                                    e.color,

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                ],

                              ),

                            ],

                          ),

                        ],

                      ),

                    )),

                    Icon(

                      Icons.arrow\_forward\_ios,

                      size: 40,

                    )

                  ],

                ),

              ),

            ));

          }).toList(),

        ));

  }

  void \_goVehicle() {}

# VehiclesController

En el **Proyecto de Visual Studio**, en la clase **UserHelper** comentamos estas líneas:

public async Task<User> GetUserAsync(string email)

{

return await \_context.Users

.Include(x => x.DocumentType)

//.Include(x => x.Vehicles)

//.ThenInclude(x => x.VehiclePhotos)

//.Include(x => x.Vehicles)

//.ThenInclude(x => x.Histories)

//.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x => x.Email == email);

}

public async Task<User> GetUserAsync(Guid id)

{

return await \_context.Users

.Include(x => x.DocumentType)

//.Include(x => x.Vehicles)

//.ThenInclude(x => x.VehiclePhotos)

//.Include(x => x.Vehicles)

//.ThenInclude(x => x.Histories)

//.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x => x.Id == id.ToString());

}

En la carpeta **Models/Request** creamos la clase **VehicleRequest**

using System;

using System.ComponentModel.DataAnnotations;

namespace Vehicles.API.Models.Request

{

public class VehicleRequest

{

public int Id { get; set; }

[Display(Name = "Tipo de vehículo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int VehicleTypeId { get; set; }

[Display(Name = "Marca")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int BrandId { get; set; }

[Display(Name = "Modelo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[Range(1900, 3000, ErrorMessage = "Valor de módelo no válido.")]

public int Model { get; set; }

[Display(Name = "Placa")]

[RegularExpression(@"[a-zA-Z]{3}[0-9]{2}[a-zA-Z0-9]", ErrorMessage = "Formato de placa incorrecto.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[StringLength(6, MinimumLength = 6, ErrorMessage = "El campo {0} debe tener {1} carácteres.")]

public string Plaque { get; set; }

[Display(Name = "Línea")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Line { get; set; }

[Display(Name = "Color")]

[MaxLength(50, ErrorMessage = "El campo {0} no puede tener más de {1} carácteres.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Color { get; set; }

[Display(Name = "Propietario")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string UserId { get; set; }

[Display(Name = "Observación")]

public string Remarks { get; set; }

public byte[] Image { get; set; }

}

}

En la carpeta **Controllers/Api** creamos la clase **VehiclesController**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.API.Models.Request;

namespace Vehicles.Api.Controllers.Api

{

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

public class VehiclesController : ControllerBase

{

private readonly DataContext \_context;

private readonly IImageHelper \_imageHelper;

private readonly IUserHelper \_userHelper;

public VehiclesController(DataContext context, IImageHelper imageHelper, IUserHelper userHelper)

{

\_context = context;

\_imageHelper = imageHelper;

\_userHelper = userHelper;

}

[HttpPost]

public async Task<ActionResult<Vehicle>> PostVehicle(VehicleRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

VehicleType vehicleType = await \_context.VehicleTypes.FindAsync(request.VehicleTypeId);

if (vehicleType == null)

{

return BadRequest("El tipo de vehículo no existe.");

}

Brand brand = await \_context.Brands.FindAsync(request.BrandId);

if (brand == null)

{

return BadRequest("La marca no existe.");

}

User user = await \_userHelper.GetUserAsync((request.UserId);

if (user == null)

{

return BadRequest("El usuario no existe.");

}

Vehicle vehicle = await \_context.Vehicles.FirstOrDefaultAsync(x => x.Plaque.ToUpper() == request.Plaque.ToUpper());

if (vehicle != null)

{

return BadRequest("Ya existe un vehículo con esa placa.");

}

string imageId = string.Empty;

List<VehiclePhoto> vehiclePhotos = new();

if (request.Image != null && request.Image.Length > 0)

{

imageId = \_imageHelper.UploadImage(request.Image, "vehicles");

vehiclePhotos.Add(new VehiclePhoto

{

ImageId = imageId

});

}

vehicle = new Vehicle

{

Brand = brand,

Color = request.Color,

Histories = new List<History>(),

Line = request.Line,

Model = request.Model,

Plaque = request.Plaque,

Remarks = request.Remarks,

User = user,

VehiclePhotos = vehiclePhotos,

VehicleType = vehicleType,

};

\_context.Vehicles.Add(vehicle);

await \_context.SaveChangesAsync();

return Ok(vehicle);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutVehicle(int id, VehicleRequest request)

{

if (id != request.Id)

{

return BadRequest();

}

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

VehicleType vehicleType = await \_context.VehicleTypes.FindAsync(request.VehicleTypeId);

if (vehicleType == null)

{

return BadRequest("El tipo de vehículo no existe.");

}

Brand brand = await \_context.Brands.FindAsync(request.BrandId);

if (brand == null)

{

return BadRequest("La marca no existe.");

}

User user = await \_userHelper.GetUserAsync(request.UserId);

if (user == null)

{

return BadRequest("El usuario no existe.");

}

Vehicle vehicle = await \_context.Vehicles.FindAsync(request.Id);

if (vehicle == null)

{

return BadRequest("El vehículo no existe.");

}

vehicle.Brand = brand;

vehicle.Color = request.Color;

vehicle.Line = request.Line;

vehicle.Model = request.Model;

vehicle.Plaque = request.Plaque;

vehicle.Remarks = request.Remarks;

vehicle.VehicleType = vehicleType;

try

{

\_context.Vehicles.Update(vehicle);

await \_context.SaveChangesAsync();

return NoContent();

}

catch (DbUpdateException dbUpdateException)

{

if (dbUpdateException.InnerException.Message.Contains("duplicate"))

{

return BadRequest("Ya existe esta marca.");

}

else

{

return BadRequest(dbUpdateException.InnerException.Message);

}

}

catch (Exception exception)

{

return BadRequest(exception.Message);

}

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteVehicle(int id)

{

Vehicle vehicle = await \_context.Vehicles

.Include(x => x.VehiclePhotos)

.Include(x => x.Histories)

.ThenInclude(x => x.Details)

.FirstOrDefaultAsync(x => x.Id == id);

if (vehicle == null)

{

return NotFound();

}

\_context.Vehicles.Remove(vehicle);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

# Agregar/Editar un Vehículo

En **user\_info\_screen.dart** hacemos:

**Cambiamos \_goAdd por \_goVehicle**

@override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(\_user.fullName),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(

                text: 'Por favor espere...',

              )

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goVehicle(Vehicle(

            brand: Brand(id: 0, description: ''),

            color: '',

            histories: [],

            historiesCount: 0,

            id: 0,

            imageFullPath: '',

            line: '',

            model: 2021,

            plaque: '',

            remarks: '',

            vehiclePhotos: [],

            vehiclePhotosCount: 0,

            vehicleType: VehicleType(id: 0, description: ''))),

      ),

    );

  }

void \_goVehicle(Vehicle vehicle) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleScreen(

                token: widget.token, user: \_user, vehicle: vehicle)));

    if (result == 'yes') {

      \_getUser();

    }

  }

En la parte de edición agregamos la **e** a **\_goVehicle()**

 Widget \_getListView() {

    return RefreshIndicator(

        onRefresh: \_getUser,

        child: ListView(

          children: \_user.vehicles.map((e) {

            return Card(

                child: InkWell(

              onTap: () => \_goVehicle(e),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Row(

                  children: <Widget>[

                    FadeInImage(

                        placeholder: AssetImage('assets/logo.png'),

                        image: NetworkImage(e.imageFullPath),

                        width: 120,

                        height: 80,

                        fit: BoxFit.cover),

                    Expanded(

En la carpeta **screens** creamos un arhcivo que llamamos **vehicle\_screen.dart**

import 'dart:io';

import 'package:camera/camera.dart';

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

import 'package:vehicles\_app/screens/take\_picture.dart';

class VehicleScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  VehicleScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_VehicleScreenState createState() => \_VehicleScreenState();

}

class \_VehicleScreenState extends State<VehicleScreen> {

  bool \_showLoader = false;

  bool \_photoChanged = false;

  late XFile \_image;

  String \_firstName = '';

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(widget.vehicle.id == 0

              ? 'Nuevo Vehículo'

              : '${widget.vehicle.brand.description} ${widget.vehicle.line} ${widget.vehicle.plaque}'),

        ),

        body: Stack(

          children: [

            SingleChildScrollView(

              child: Column(

                children: <Widget>[

                  \_showPhoto(),

                  \_showVehicleType(),

                  \_showBrand(),

                  \_showModel(),

                  \_showPlaque(),

                  \_showLine(),

                  \_showColor(),

                  \_showRemarks(),

                  \_showButtons(),

                ],

              ),

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showPhoto() {

    return InkWell(

      child: Stack(children: <Widget>[

        Container(

          margin: EdgeInsets.only(top: 10),

          child: widget.vehicle.id == 0 && !\_photoChanged

              ? Image(

                  image: AssetImage('assets/noimage.png'),

                  width: 160,

                  height: 160,

                  fit: BoxFit.cover)

              : ClipRRect(

                  borderRadius: BorderRadius.circular(80),

                  child: \_photoChanged

                      ? Image.file(

                          File(\_image.path),

                          width: 160,

                          height: 160,

                          fit: BoxFit.cover,

                        )

                      : FadeInImage(

                          placeholder: AssetImage('assets/logo.png'),

                          image: NetworkImage(widget.vehicle.imageFullPath),

                          width: 160,

                          height: 160,

                          fit: BoxFit.cover,

                        ),

                ),

        ),

        Positioned(

            bottom: 0,

            left: 100,

            child: InkWell(

              onTap: () => \_takePicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.photo\_camera,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

        Positioned(

            bottom: 0,

            left: 0,

            child: InkWell(

              onTap: () => \_selectPicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.image,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

      ]),

    );

  }

  Widget \_showVehicleType() {

    return Container();

  }

  Widget \_showBrand() {

    return Container();

  }

  Widget \_showModel() {

    return Container();

  }

  Widget \_showPlaque() {

    return Container();

  }

  Widget \_showLine() {

    return Container();

  }

  Widget \_showColor() {

    return Container();

  }

  Widget \_showRemarks() {

    return Container();

  }

  Widget \_showButtons() {

    return Container();

  }

  void \_takePicture() async {

    WidgetsFlutterBinding.ensureInitialized();

    final cameras = await availableCameras();

    final firstCamera = cameras.first;

    Response? response = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => TakePictureScreen(

                  camera: firstCamera,

                )));

    if (response != null) {

      setState(() {

        \_photoChanged = true;

        \_image = response.result;

      });

    }

  }

  void \_selectPicture() async {

    final ImagePicker \_picker = ImagePicker();

    final XFile? image = await \_picker.pickImage(source: ImageSource.gallery);

    if (image != null) {

      setState(() {

        \_photoChanged = true;

        \_image = image;

      });

    }

  }

}

Hasta aquí navegamos a la pantalla de ver un vehículo pero solo vemos la foto principal,

A continuación agregamos el resto de los campos y hacemos un carrousel de fotos.

# Mejorar rendimiento de imágenes

Vamos a la página

<https://pub.dev/>

Buscamos **cached network\_image**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

cached\_network\_image: ^3.1.0

Copiamos la línea

cached\_network\_image: ^3.1.0

y la pegamos en el archivo **pubspec.yaml**

Cambiamops todos los **FadeInImage** por **CachedNetworkImage**

En **vehicle\_screen.dart** hacemos:

import 'dart:convert';

import 'dart:io';

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:cached\_network\_image/cached\_network\_image.dart';

import 'package:camera/camera.dart';

import 'package:connectivity/connectivity.dart';

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/brand.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

import 'package:vehicles\_app/models/vehicle\_type.dart';

import 'package:vehicles\_app/screens/take\_picture.dart';

class VehicleScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  VehicleScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_VehicleScreenState createState() => \_VehicleScreenState();

}

class \_VehicleScreenState extends State<VehicleScreen> {

  bool \_showLoader = false;

  bool \_photoChanged = false;

  late XFile \_image;

  int \_vehicleTypeId = 0;

  String \_vehicleTypeIdError = '';

  bool \_vehicleTypeIdShowError = false;

  List<VehicleType> \_vehicleTypes = [];

  int \_brandId = 0;

  String \_brandIdError = '';

  bool \_brandIdShowError = false;

  List<Brand> \_brands = [];

  String \_line = '';

  String \_lineError = '';

  bool \_lineShowError = false;

  TextEditingController \_lineController = TextEditingController();

  String \_color = '';

  String \_colorError = '';

  bool \_colorShowError = false;

  TextEditingController \_colorController = TextEditingController();

  String \_model = '';

  String \_modelError = '';

  bool \_modelShowError = false;

  TextEditingController \_modelController = TextEditingController();

  String \_plaque = '';

  String \_plaqueError = '';

  bool \_plaqueShowError = false;

  TextEditingController \_plaqueController = TextEditingController();

  String \_remarks = '';

  String \_remarksError = '';

  bool \_remarksShowError = false;

  TextEditingController \_remarksController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_getVehiclesTypes();

    \_getBrands();

    \_loadFieldValues();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(widget.vehicle.id == 0

              ? 'Nuevo Vehículo'

              : '${widget.vehicle.brand.description} ${widget.vehicle.line} ${widget.vehicle.plaque}'),

        ),

        body: Stack(

          children: [

            SingleChildScrollView(

              child: Column(

                children: <Widget>[

                  \_showPhoto(),

                  \_showVehicleType(),

                  \_showBrand(),

                  \_showLine(),

                  \_showColor(),

                  \_showModel(),

                  \_showPlaque(),

                  \_showRemarks(),

                  \_showButtons(),

                ],

              ),

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showPhoto() {

    return InkWell(

      child: Stack(children: <Widget>[

        Container(

          margin: EdgeInsets.only(top: 10),

          child: widget.vehicle.id == 0 && !\_photoChanged

              ? Image(

                  image: AssetImage('assets/noimage.png'),

                  width: 160,

                  height: 160,

                  fit: BoxFit.cover)

              : ClipRRect(

                  borderRadius: BorderRadius.circular(10),

                  child: \_photoChanged

                      ? Image.file(

                          File(\_image.path),

                          width: 160,

                          height: 160,

                          fit: BoxFit.cover,

                        )

                      : CachedNetworkImage(

                          imageUrl: widget.vehicle.imageFullPath,

                          errorWidget: (context, url, error) =>

                              Icon(Icons.error),

                          fit: BoxFit.cover,

                          height: 160,

                          width: 160,

                          placeholder: (context, url) => Image(

                            image: AssetImage('assets/vehicles\_logo.png'),

                            fit: BoxFit.cover,

                            height: 160,

                            width: 160,

                          ),

                        ),

                ),

        ),

        Positioned(

            bottom: 0,

            left: 100,

            child: InkWell(

              onTap: () => \_takePicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.photo\_camera,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

        Positioned(

            bottom: 0,

            left: 0,

            child: InkWell(

              onTap: () => \_selectPicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.image,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

      ]),

    );

  }

  Widget \_showVehicleType() {

    return Container(

      padding: EdgeInsets.all(10),

      child: \_vehicleTypes.length == 0

          ? Text('Cargando tipos de vehículo...')

          : DropdownButtonFormField(

              items: \_getComboVehicleTypes(),

              value: \_vehicleTypeId,

              onChanged: (option) {

                setState(() {

                  \_vehicleTypeId = option as int;

                });

              },

              decoration: InputDecoration(

                hintText: 'Seleccione un tipo de vehículo...',

                labelText: 'Tipo de Vehículo',

                errorText: \_vehicleTypeIdShowError ? \_vehicleTypeIdError : null,

                border:

                    OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

              )),

    );

  }

  Widget \_showBrand() {

    return Container(

      padding: EdgeInsets.all(10),

      child: \_brands.length == 0

          ? Text('Cargando marcas de vehículos...')

          : DropdownButtonFormField(

              items: \_getComboBrands(),

              value: \_brandId,

              onChanged: (option) {

                setState(() {

                  \_brandId = option as int;

                });

              },

              decoration: InputDecoration(

                hintText: 'Seleccione una marca de vehículo...',

                labelText: 'Marca de Vehículo',

                errorText: \_brandIdShowError ? \_brandIdError : null,

                border:

                    OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

              )),

    );

  }

  Widget \_showLine() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_lineController,

        decoration: InputDecoration(

            hintText: 'Ingresa línea...',

            labelText: 'Línea',

            errorText: \_lineShowError ? \_lineError : null,

            suffixIcon: Icon(Icons.directions\_car\_filled),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_line = value;

        },

      ),

    );

  }

  Widget \_showColor() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_colorController,

        decoration: InputDecoration(

            hintText: 'Ingresa color...',

            labelText: 'Color',

            errorText: \_colorShowError ? \_colorError : null,

            suffixIcon: Icon(Icons.palette),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_color = value;

        },

      ),

    );

  }

  Widget \_showModel() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType: TextInputType.number,

        controller: \_modelController,

        decoration: InputDecoration(

            hintText: 'Ingresa modelo...',

            labelText: 'Modelo',

            errorText: \_modelShowError ? \_modelError : null,

            suffixIcon: Icon(Icons.event),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_model = value;

        },

      ),

    );

  }

  Widget \_showPlaque() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_plaqueController,

        decoration: InputDecoration(

            hintText: 'Ingresa patente...',

            labelText: 'Patente',

            errorText: \_plaqueShowError ? \_plaqueError : null,

            suffixIcon: Icon(Icons.money),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_plaque = value;

        },

      ),

    );

  }

  Widget \_showRemarks() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_remarksController,

        decoration: InputDecoration(

            hintText: 'Ingresa comentarios...',

            labelText: 'Comentarios',

            errorText: \_remarksShowError ? \_remarksError : null,

            suffixIcon: Icon(Icons.notes),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_remarks = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.vehicle.id == 0

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.vehicle.id == 0

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

  void \_takePicture() async {

    WidgetsFlutterBinding.ensureInitialized();

    final cameras = await availableCameras();

    final firstCamera = cameras.first;

    Response? response = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => TakePictureScreen(

                  camera: firstCamera,

                )));

    if (response != null) {

      setState(() {

        \_photoChanged = true;

        \_image = response.result;

      });

    }

  }

  void \_selectPicture() async {

    final ImagePicker \_picker = ImagePicker();

    final XFile? image = await \_picker.pickImage(source: ImageSource.gallery);

    if (image != null) {

      setState(() {

        \_photoChanged = true;

        \_image = image;

      });

    }

  }

  List<DropdownMenuItem<int>> \_getComboVehicleTypes() {

    List<DropdownMenuItem<int>> list = [];

    list.add(DropdownMenuItem(

      child: Text('Seleccione un Tipo de Vehículo...'),

      value: 0,

    ));

    \_vehicleTypes.forEach((vehicleType) {

      list.add(DropdownMenuItem(

        child: Text(vehicleType.description),

        value: vehicleType.id,

      ));

    });

    return list;

  }

  Future<Null> \_getVehiclesTypes() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.getVehicleTypes(widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_vehicleTypes = response.result;

    });

  }

  List<DropdownMenuItem<int>> \_getComboBrands() {

    List<DropdownMenuItem<int>> list = [];

    list.add(DropdownMenuItem(

      child: Text('Seleccione una Marca de Vehículo...'),

      value: 0,

    ));

    \_brands.forEach((brandType) {

      list.add(DropdownMenuItem(

        child: Text(brandType.description),

        value: brandType.id,

      ));

    });

    return list;

  }

  Future<Null> \_getBrands() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.getBrands(widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_brands = response.result;

    });

  }

  \_save() {

    if (!validateFields()) {

      return;

    }

    widget.vehicle.id == 0 ? \_addRecord() : \_saveRecord();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_vehicleTypeId == 0) {

      isValid = false;

      \_vehicleTypeIdShowError = true;

      \_vehicleTypeIdError = 'Debes seleccionar un Tipo de Vehículo';

    } else {

      \_vehicleTypeIdShowError = false;

    }

    if (\_brandId == 0) {

      isValid = false;

      \_brandIdShowError = true;

      \_brandIdError = 'Debes seleccionar una Marca de Vehículo';

    } else {

      \_brandIdShowError = false;

    }

    if (\_line.isEmpty) {

      isValid = false;

      \_lineShowError = true;

      \_lineError = 'Debes ingresar una Línea';

    } else {

      \_lineShowError = false;

    }

    if (\_color.isEmpty) {

      isValid = false;

      \_colorShowError = true;

      \_colorError = 'Debes ingresar un color';

    } else {

      \_colorShowError = false;

    }

    if (\_model.isEmpty) {

      isValid = false;

      \_modelShowError = true;

      \_modelError = 'Debes ingresar un modelo';

    } else {

      int model = int.parse(\_model);

      if (model < 1900 || model > 3000) {

        isValid = false;

        \_modelShowError = true;

        \_modelError = 'El modelo debe ser un n° entre 1900 y 3000';

      } else {

        \_modelShowError = false;

      }

    }

    if (\_plaque.isEmpty) {

      isValid = false;

      \_plaqueShowError = true;

      \_plaqueError = 'Debes ingresar una Patente';

    } else {

      if (!RegExp('[a-zA-Z]{3}[0-9]{2}[a-zA-Z0-9]').hasMatch(\_plaque)) {

        isValid = false;

        \_plaqueShowError = true;

        \_plaqueError = 'El formato de la placa es incorrecto.';

      } else {

        \_plaqueShowError = false;

      }

    }

    setState(() {});

    return isValid;

  }

  void \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    String base64Image = '';

    if (\_photoChanged) {

      List<int> imageBytes = await \_image.readAsBytes();

      base64Image = base64Encode(imageBytes);

    }

    Map<String, dynamic> request = {

      'vehicleTypeId': \_vehicleTypeId,

      'brandId': \_brandId,

      'model': \_model,

      'plaque': \_plaque.toUpperCase(),

      'line': \_line,

      'color': \_color,

      'userId': widget.user.id,

      'remarks': \_remarks,

      'image': base64Image,

    };

    Response response =

        await ApiHelper.post('/api/Vehicles/', request, widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Map<String, dynamic> request = {

      'id': widget.vehicle.id,

      'vehicleTypeId': \_vehicleTypeId,

      'brandId': \_brandId,

      'model': \_model,

      'plaque': \_plaque.toUpperCase(),

      'line': \_line,

      'color': \_color,

      'userId': widget.user.id,

      'remarks': \_remarks,

    };

    Response response = await ApiHelper.put(

        '/api/Vehicles/', widget.vehicle.id.toString(), request, widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estas seguro de querer borrar el vehículo?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.delete(

        '/api/Vehicles/', widget.vehicle.id.toString(), widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_loadFieldValues() {

    \_vehicleTypeId = widget.vehicle.vehicleType.id;

    \_brandId = widget.vehicle.brand.id;

    \_model = widget.vehicle.model.toString();

    \_modelController.text = \_model;

    \_plaque = widget.vehicle.plaque;

    \_plaqueController.text = \_plaque;

    \_line = widget.vehicle.line;

    \_lineController.text = \_line;

    \_color = widget.vehicle.color;

    \_colorController.text = \_color;

    \_remarks = widget.vehicle.remarks == null ? '' : widget.vehicle.remarks!;

    \_remarksController.text = \_remarks;

  }

}

# Carousel de imágenes

Vamos a la página

<https://pub.dev/>

Buscamos **carousel\_slider**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

carousel\_slider: ^4.0.0

Copiamos la línea

carousel\_slider: ^4.0.0

y la pegamos en el archivo **pubspec.yaml**

Al método **\_showPhoto** lo renombramos como **\_showUniquePhoto** y creamos un nuevo método **\_showPhoto** que llamará al **\_showUniquePhoto** cuando sea un nuevo vehículo o llamará a un nuevo método que llamaremos **\_showPhotosCarousel** cuando sea un vehículo existente:

Widget \_showPhoto() {

    return widget.vehicle.id == 0 ? \_showUniquePhoto() : \_showPhotosCarousel();

  }

  Widget \_showUniquePhoto() {

    return InkWell(

      child: Stack(children: <Widget>[

        Container(

Al principio ponemos:

@override

  void initState() {

    super.initState();

    \_loadData();

  }

Y al final ponemos el método **\_loadData** (es para hacer que espere la carga de VehiclesTypes y Brands para los combobox):

void \_loadData() async {

    await \_getVehiclesTypes();

    await \_getBrands();

    \_loadFieldValues();

  }

Agregamos estas variables:

class \_VehicleScreenState extends State<VehicleScreen> {

  bool \_showLoader = false;

  bool \_photoChanged = false;

  late XFile \_image;

  int \_current = 0;

  CarouselController \_carouselController = CarouselController();

Y ponemos el método **\_showPhotosCarousel**

Widget \_showPhotosCarousel() {

    return Container(

      margin: EdgeInsets.symmetric(vertical: 20),

      child: Column(

        children: [

          CarouselSlider(

            options: CarouselOptions(

                height: 200,

                autoPlay: true,

                autoPlayInterval: Duration(seconds: 3),

                enlargeCenterPage: true,

                onPageChanged: (index, reason) {

                  setState(() {

                    \_current = index;

                  });

                }),

            carouselController: \_carouselController,

            items: widget.vehicle.vehiclePhotos.map((i) {

              return Builder(

                builder: (BuildContext context) {

                  return Container(

                      width: MediaQuery.of(context).size.width,

                      margin: EdgeInsets.symmetric(horizontal: 5),

                      child: ClipRRect(

                        borderRadius: BorderRadius.circular(20),

                        child: CachedNetworkImage(

                          imageUrl: i.imageFullPath,

                          errorWidget: (context, url, error) =>

                              Icon(Icons.error),

                          fit: BoxFit.cover,

                          height: 300,

                          width: 300,

                          placeholder: (context, url) => Image(

                            image: AssetImage('assets/logo.png'),

                            fit: BoxFit.cover,

                            height: 300,

                            width: 300,

                          ),

                        ),

                      ));

                },

              );

            }).toList(),

          ),

          Row(

            mainAxisAlignment: MainAxisAlignment.center,

            children: widget.vehicle.vehiclePhotos.asMap().entries.map((entry) {

              return GestureDetector(

                onTap: () => \_carouselController.animateToPage(entry.key),

                child: Container(

                  width: 12.0,

                  height: 12.0,

                  margin: EdgeInsets.symmetric(vertical: 8.0, horizontal: 4.0),

                  decoration: BoxDecoration(

                      shape: BoxShape.circle,

                      color: (Theme.of(context).brightness == Brightness.dark

                              ? Colors.white

                              : Colors.black)

                          .withOpacity(\_current == entry.key ? 0.9 : 0.4)),

                ),

              );

            }).toList(),

          ),

          \_showImageButtons()

        ],

      ),

    );

  }

Widget \_showImageButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Row(

                mainAxisAlignment: MainAxisAlignment.spaceAround,

                children: [

                  Icon(Icons.add\_a\_photo),

                  Text('Adicionar Foto'),

                ],

              ),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_goAddPhoto(),

            ),

          ),

          SizedBox(

            width: 20,

          ),

          Expanded(

            child: ElevatedButton(

              child: Row(

                mainAxisAlignment: MainAxisAlignment.spaceAround,

                children: [

                  Icon(Icons.delete),

                  Text('Eliminar Foto'),

                ],

              ),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFFB4161B);

                }),

              ),

              onPressed: () => \_confirmDeletePhoto(),

            ),

          ),

        ],

      ),

    );

  }

  \_goAddPhoto() {}

  \_confirmDeletePhoto() {}

# Borrar y Agregar foto al Carousel

En el Proyecto de Visual Studio en la carpeta Models/Request creamos la clase **VehiclePhotoRequest**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.API.Models.Request

{

public class VehiclePhotoRequest

{

public int Id { get; set; }

[Required]

public int VehicleId { get; set; }

[Required]

public byte[] Image { get; set; }

}

}

Creamos en la carpeta **Controllers/API** el controlador **VehiclePhotosController**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.API.Models.Request;

namespace Vehicles.API.Controllers.API

{

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

public class VehiclePhotosController : ControllerBase

{

private readonly DataContext \_context;

private readonly IImageHelper \_imageHelper;

public VehiclePhotosController(DataContext context, IImageHelper imageHelper)

{

\_context = context;

\_imageHelper = imageHelper;

}

[HttpPost]

public async Task<ActionResult<VehiclePhoto>> PostVehiclePhoto(VehiclePhotoRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

Vehicle vehicle = await \_context.Vehicles.FindAsync(request.VehicleId);

if (vehicle == null)

{

return BadRequest("El vehículo no existe.");

}

string imageId = \_imageHelper.UploadImage(request.Image, "vehicles");

VehiclePhoto vehiclePhoto = new()

{

ImageId = imageId,

Vehicle = vehicle

};

\_context.VehiclePhotos.Add(vehiclePhoto);

await \_context.SaveChangesAsync();

return Ok(vehicle);

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteVehiclePhoto(int id)

{

VehiclePhoto vehiclePhoto = await \_context.VehiclePhotos.FindAsync(id);

if (vehiclePhoto == null)

{

return NotFound();

}

\_context.VehiclePhotos.Remove(vehiclePhoto);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

En el archivo **vehicle\_screen.dart** cambiamos:

Future<Null> \_takePicture() async {

Future<Null> \_selectPicture() async {

Creamos los métodos **\_goAddPhoto**, **\_confirmDeletePhoto**, **\_addPicture** y **\_deletePhoto**

void \_goAddPhoto() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿De donde deseas obtener la imagen?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'cancel', label: 'Cancelar'),

          AlertDialogAction(key: 'camera', label: 'Cámara'),

          AlertDialogAction(key: 'gallery', label: 'Galería'),

        ]);

    if (response == 'cancel') {

      return;

    }

    if (response == 'camera') {

      await \_takePicture();

    } else {

      await \_selectPicture();

    }

    if (\_photoChanged) {

      \_addPicture();

    }

  }

  void \_confirmDeletePhoto() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estas seguro de querer borrar la última foto tomada?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deletePhoto();

    }

  }

  void \_addPicture() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    List<int> imageBytes = await \_image.readAsBytes();

    String base64Image = base64Encode(imageBytes);

    Map<String, dynamic> request = {

      'vehicleId': widget.vehicle.id,

      'image': base64Image

    };

    Response response =

        await ApiHelper.post('/api/VehiclePhotos', request, widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_deletePhoto() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.delete(

        '/api/VehiclePhotos/',

        widget.vehicle.vehiclePhotos[widget.vehicle.vehiclePhotos.length - 1].id

            .toString(),

        widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

# Vehicle Info Screen

Creamos el archivo **vehicle\_info\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

class VehicleInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  VehicleInfoScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_VehicleInfoScreenState createState() => \_VehicleInfoScreenState();

}

class \_VehicleInfoScreenState extends State<VehicleInfoScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text('Vehicle Info'),

        ),

        body: Center(

          child: Text('Vehicle Info'),

        ));

  }

}

Modificamos en **user\_info\_screen**

floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAddVehicle(Vehicle(

void \_goVehicle(Vehicle vehicle) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleInfoScreen(

                token: widget.token, user: \_user, vehicle: vehicle)));

    if (result == 'yes') {

      \_getUser();

    }

  }

void \_goVehicle(Vehicle vehicle) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleInfoScreen(

                token: widget.token, user: \_user, vehicle: vehicle)));

    if (result == 'yes') {

      \_getUser();

    }

  }

  void \_goAddVehicle(Vehicle vehicle) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleScreen(

                token: widget.token, user: \_user, vehicle: vehicle)));

    if (result == 'yes') {

      \_getUser();

    }

  }

Creamos el archivo **historyScreen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

class HistoryScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  HistoryScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_HistoryScreenState createState() => \_HistoryScreenState();

}

class \_HistoryScreenState extends State<HistoryScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title:Text('History'),

      ),

      body:Center(

        child: Text('History'),

      )

    );

  }

}

En **vehicle\_info\_screen.dart** hacemos:

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

import 'package:vehicles\_app/screens/history\_screen.dart';

class VehicleInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  VehicleInfoScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_VehicleInfoScreenState createState() => \_VehicleInfoScreenState();

}

class \_VehicleInfoScreenState extends State<VehicleInfoScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

            '${widget.vehicle.brand.description} ${widget.vehicle.line} ${widget.vehicle.plaque}'),

      ),

  body: Center(

       child: \_showLoader

          ? LoaderComponent(text: 'Por favor espere...',)

          : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAddHistory(History(

            date: '',

            dateLocal: '',

            details: [],

            detailsCount: 0,

            id: 0,

            mileage: 0,

            remarks: '',

            total: 0,

            totalLabor: 0,

            totalSpareParts: 0)),

      ),

    );

  }

  void \_goAddHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryScreen(

                token: widget.token,

                user: widget.user,

                vehicle: widget.vehicle)));

    if (result == 'yes') {

      //TODO: pending fresh the page

    }

  }

}

Completamos parcialmente la **vehicle\_info\_screen.dart**:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:cached\_network\_image/cached\_network\_image.dart';

import 'package:connectivity/connectivity.dart';

import 'package:flutter/material.dart';

import 'package:intl/intl.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

import 'package:vehicles\_app/screens/history\_screen.dart';

import 'package:vehicles\_app/screens/vehicle\_screen.dart';

class VehicleInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  VehicleInfoScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_VehicleInfoScreenState createState() => \_VehicleInfoScreenState();

}

class \_VehicleInfoScreenState extends State<VehicleInfoScreen> {

  bool \_showLoader = false;

  late Vehicle \_vehicle;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

            '${widget.vehicle.brand.description} ${widget.vehicle.line} ${widget.vehicle.plaque}'),

      ),

      body: Center(

        child: \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goHistory(History(

            date: '',

            dateLocal: '',

            details: [],

            detailsCount: 0,

            id: 0,

            mileage: 0,

            remarks: '',

            total: 0,

            totalLabor: 0,

            totalSpareParts: 0)),

      ),

    );

  }

  void \_goHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryScreen(

                token: widget.token,

                user: widget.user,

                vehicle: widget.vehicle)));

    if (result == 'yes') {

      //TODO: pending fresh the page

    }

  }

  Widget \_getContent() {

    return Column(

      children: <Widget>[

        \_showVehicleInfo(),

        Expanded(

          child: widget.vehicle.histories.length == 0

              ? \_noContent()

              : \_getListView(),

        )

      ],

    );

  }

  Widget \_showVehicleInfo() {

    return Container(

      margin: EdgeInsets.all(10),

      padding: EdgeInsets.all(5),

      child: Row(

        children: <Widget>[

          Stack(

            children: <Widget>[

              ClipRRect(

                borderRadius: BorderRadius.circular(10),

                child: CachedNetworkImage(

                  imageUrl: widget.vehicle.imageFullPath,

                  errorWidget: (context, url, error) => Icon(Icons.error),

                  fit: BoxFit.cover,

                  height: 100,

                  width: 140,

                  placeholder: (context, url) => Image(

                    image: AssetImage('assets/logo.png'),

                    fit: BoxFit.cover,

                    height: 300,

                    width: 300,

                  ),

                ),

              ),

              Positioned(

                  bottom: 0,

                  left: 100,

                  child: InkWell(

                    onTap: () => \_goEdit(),

                    child: ClipRRect(

                      borderRadius: BorderRadius.circular(30),

                      child: Container(

                        color: Colors.green[50],

                        height: 40,

                        width: 40,

                        child: Icon(

                          Icons.edit,

                          size: 30,

                          color: Colors.blue,

                        ),

                      ),

                    ),

                  )),

            ],

          ),

          Expanded(

            child: Container(

              margin: EdgeInsets.symmetric(horizontal: 10),

              child: Row(

                mainAxisAlignment: MainAxisAlignment.start,

                children: [

                  Expanded(

                    child: Column(

                      mainAxisAlignment: MainAxisAlignment.start,

                      children: <Widget>[

                        Row(

                          children: <Widget>[

                            Text('Tipo de vehículo: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.vehicleType.description,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('Marca: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.brand.description,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('Modelo: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.model.toString(),

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('Placa: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.plaque,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('Línea: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.line,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('Color: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.color,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('Comentarios: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.remarks == null

                                  ? 'N/A'

                                  : widget.vehicle.remarks!,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(

                          height: 5,

                        ),

                        Row(

                          children: <Widget>[

                            Text('N° de Historias: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              widget.vehicle.historiesCount.toString(),

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                      ],

                    ),

                  ),

                ],

              ),

            ),

          ),

        ],

      ),

    );

  }

  Widget \_noContent() {

    return Center(

      child: Container(

        margin: EdgeInsets.all(20),

        child: Text(

          'El vehículo no tiene historias registradas.',

          style: TextStyle(

            fontSize: 16,

            fontWeight: FontWeight.bold,

          ),

        ),

      ),

    );

  }

  \_getListView() {

    return RefreshIndicator(

        onRefresh: \_getVehicle,

        child: ListView(

          children: widget.vehicle.histories.map((e) {

            return Card(

                child: InkWell(

              onTap: () => \_goHistory(e),

              child: Container(

                margin: EdgeInsets.all(10),

                padding: EdgeInsets.all(5),

                child: Row(

                  children: <Widget>[

                    Expanded(

                        child: Container(

                      margin: EdgeInsets.symmetric(horizontal: 10),

                      child: Row(

                        mainAxisAlignment: MainAxisAlignment.start,

                        children: <Widget>[

                          Column(

                            children: <Widget>[

                              Text(

                                '${DateFormat('dd/MM/yyyy').format(DateTime.parse(e.dateLocal))}',

                                style: TextStyle(

                                    fontSize: 14, fontWeight: FontWeight.bold),

                              ),

                              Row(

                                children: [

                                  Text(

                                    '${e.mileage} km',

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                  SizedBox(

                                    width: 5,

                                  ),

                                  Text(

                                    e.remarks == null ? 'N/A' : e.remarks!,

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                ],

                              ),

                              Row(

                                children: [

                                  Text(

                                    'Mano de Obra: ${NumberFormat.currency(symbol: '\$').format(e.totalLabor)}',

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                  SizedBox(

                                    width: 5,

                                  ),

                                ],

                              ),

                              Row(

                                children: [

                                  Text(

                                    'Repuestos: ${NumberFormat.currency(symbol: '\$').format(e.totalSpareParts)}',

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                ],

                              ),

                              Row(

                                children: [

                                  Text(

                                    'Total: ${NumberFormat.currency(symbol: '\$').format(e.total)}',

                                    style: TextStyle(

                                      fontSize: 14,

                                    ),

                                  ),

                                ],

                              ),

                            ],

                          ),

                        ],

                      ),

                    )),

                    Icon(

                      Icons.arrow\_forward\_ios,

                      size: 40,

                    )

                  ],

                ),

              ),

            ));

          }).toList(),

        ));

  }

  void \_goEdit() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: widget.vehicle,

                )));

    if (result == 'yes') {

      //TODO: Pending refresh vehicle info

    }

  }

  Future<Null> \_getVehicle() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    //TODO:Pending to get the vehicle

    Response response = await ApiHelper.getUser(widget.token, widget.user.id);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    // setState(() {

    //   \_user = response.result;

    // });

  }

}

Vamos al Proyecto de Visual Studio y agregamos al Controlador Api **VehiclesController**

// GET: api/Vehicles/5

[HttpGet("{id}")]

public async Task<ActionResult<Vehicle>> GetVehicle(int id)

{

Vehicle vehicle = await \_context.Vehicles

.Include(x=>x.VehicleType)

.Include(x => x.Brand)

.Include(x => x.VehiclePhotos)

.Include(x => x.Histories)

.ThenInclude(x => x.Details)

.ThenInclude(x => x.Procedure)

.FirstOrDefaultAsync(x=> x.Id==id);

if (vehicle == null)

{

return NotFound();

}

return vehicle;

}

En **api\_helper.dart** agregamos el método **getVehicle**:

 static Future<Response> getVehicle(Token token,String id) async {

    if (!\_validateToken(token)) {

      return Response(

          isSuccess: false,

          message:

              'Sus credenciales se han vencido, por favor cierre sesión y vuelva a ingresar al sistema.');

    }

    var url = Uri.parse('${Constants.apiUrl}/api/Vehicles/$id');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${token.token}',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    var decodedJson = jsonDecode(body);

    return Response(isSuccess: true, result: Vehicle.fromJson(decodedJson));

  }

En **vehicle\_info\_screen.dart** hacemos:

class VehicleInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  VehicleInfoScreen(

      {required this.token, required this.user, required this.vehicle});

  @override

  \_VehicleInfoScreenState createState() => \_VehicleInfoScreenState();

}

class \_VehicleInfoScreenState extends State<VehicleInfoScreen> {

  bool \_showLoader = false;

  late Vehicle \_vehicle;

  @override

  void initState() {

    super.initState();

    \_vehicle = widget.vehicle;

  }

Response response =

        await ApiHelper.getVehicle(widget.token, widget.vehicle.id.toString());

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_vehicle = response.result;

    });

Reemplazamos todolo que sea **widget.vehicle** por **\_vehicle**

excepto este:

@override

  void initState() {

    super.initState();

    \_vehicle = widget.vehicle;

  }

Modificamos en **\_goHistory**

 void \_goHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryScreen(

                token: widget.token, user: widget.user, vehicle: \_vehicle)));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

Modificamos en **\_goEdit**

void \_goEdit() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: \_vehicle,

                )));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

# Add History y Edit History

En el Proyecto de Visual Studio en la carpeta Models/Request creamos la clase **HistoryRequest**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.API.Models.Request

{

public class HistoryRequest

{

public int Id { get; set; }

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int VehicleId { get; set; }

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[DisplayFormat(DataFormatString = "{0:N0}")]

public int Mileage { get; set; }

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public string Remarks { get; set; }

}

}

Creamos en la carpeta **Controllers/API** el controlador **HistoriesController**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.Api.Helpers;

using Vehicles.API.Models.Request;

namespace Vehicles.API.Controllers.API

{

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

public class HistoriesController : ControllerBase

{

private readonly DataContext \_context;

private readonly IUserHelper \_userHelper;

public HistoriesController(DataContext context, IUserHelper userHelper)

{

\_context = context;

\_userHelper = userHelper;

}

[HttpGet("{id}")]

public async Task<ActionResult<History>> GetHistory(int id)

{

History history = await \_context.Histories

.Include(x => x.Details)

.ThenInclude(x => x.Procedure)

.FirstOrDefaultAsync(x => x.Id == id);

if (history == null)

{

return NotFound();

}

return history;

}

[HttpPost]

public async Task<ActionResult<History>> PostHistory(HistoryRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

Vehicle vehicle = await \_context.Vehicles.FindAsync(request.VehicleId);

if (vehicle == null)

{

return BadRequest("El vehículo no existe.");

}

string email = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier).Value;

User user = await \_userHelper.GetUserAsync(email);

if (user == null)

{

return BadRequest("El usuario no existe.");

}

History history = new()

{

Date = DateTime.UtcNow,

Details = new List<Detail>(),

Mileage = request.Mileage,

Remarks = request.Remarks,

User = user,

Vehicle = vehicle,

};

\_context.Histories.Add(history);

await \_context.SaveChangesAsync();

return Ok(history);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutHistory(int id, HistoryRequest request)

{

if (id != request.Id)

{

return BadRequest();

}

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

History history = await \_context.Histories.FindAsync(request.Id);

if (history == null)

{

return BadRequest("La historia no existe.");

}

history.Mileage = request.Mileage;

history.Remarks = request.Remarks;

\_context.Histories.Update(history);

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteHistory(int id)

{

History history = await \_context.Histories

.Include(x => x.Details)

.FirstOrDefaultAsync(x => x.Id == id);

if (history == null)

{

return NotFound();

}

\_context.Histories.Remove(history);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

En **api\_helper.dart** agregamos el método **getHistory**:

  static Future<Response> getHistory(Token token, String id) async {

    if (!\_validateToken(token)) {

      return Response(

          isSuccess: false,

          message:

              'Sus credenciales se han vencido, por favor cierre sesión y vuelva a ingresar al sistema.');

    }

    var url = Uri.parse('${Constants.apiUrl}/api/Histories/$id');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

        'authorization': 'bearer ${token.token}',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

    var decodedJson = jsonDecode(body);

    return Response(isSuccess: true, result: History.fromJson(decodedJson));

  }

En **vehicle\_info\_screen** agregamos al método **\_goHistory**:

void \_goHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: \_vehicle,

                  history: history,

                )));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

En el Proyecto de Visual Studio cambiamos en la Entity Detail

using System.ComponentModel.DataAnnotations;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class Detail

{

public int Id { get; set; }

[JsonIgnore]

[Display(Name = "Historia")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public History History { get; set; }

[Display(Name = "Procedimiento")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Procedure Procedure { get; set; }

[Display(Name = "Precio Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int LaborPrice { get; set; }

[Display(Name = "Precio Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int SparePartsPrice { get; set; }

[Display(Name = "Total")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public int TotalPrice => LaborPrice + SparePartsPrice;

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

}

}

Y cambiamos en la Entity **History**

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Text.Json.Serialization;

namespace Vehicles.Api.Data.Entities

{

public class History

{

public int Id { get; set; }

[JsonIgnore]

[Display(Name = "Vehículo")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public Vehicle Vehicle { get; set; }

[Display(Name = "Fecha")]

[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

public DateTime Date { get; set; }

[Display(Name = "Fecha")]

[DisplayFormat(DataFormatString = "{0:yyyy/MM/dd hh:mm tt}")]

public DateTime DateLocal => Date.ToLocalTime();

[Display(Name = "Kilometraje")]

[DisplayFormat(DataFormatString = "{0:N0}")]

public int Mileage { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

[JsonIgnore]

[Display(Name = "Mecánico")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public User User { get; set; }

public ICollection<Detail> Details { get; set; }

[Display(Name = "N° Detalles")]

public int DetailsCount => Details == null ? 0 : Details.Count;

[Display(Name = "Total Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public int TotalLabor => Details == null ? 0 : Details.Sum(x => x.LaborPrice);

[Display(Name = "Total Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public int TotalSpareParts => Details == null ? 0 : Details.Sum(x => x.SparePartsPrice);

[Display(Name = "Total")]

[DisplayFormat(DataFormatString = "{0:C2}")]

public int Total => Details == null ? 0 : Details.Sum(x => x.TotalPrice);

}

}

En **DetailViewModel** cambiamos:

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using Vehicles.Api.Data.Entities;

namespace Vehicles.Api.Models

{

public class DetailViewModel

{

public int Id { get; set; }

[Display(Name = "Precio Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int LaborPrice { get; set; }

[Display(Name = "Precio Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int SparePartsPrice { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

public int HistoryId { get; set; }

[Display(Name = "Procedimiento")]

[Range(1, int.MaxValue, ErrorMessage = "Debes seleccionar un procedimiento.")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int ProcedureId { get; set; }

public IEnumerable<SelectListItem> Procedures { get; set; }

}

}

Corremos en la Consola del Administrador de Paquetes el comando

add-migration DecimalToInt

update-database

Si no funciona, cambio en la base de datos los campos LaborPrice y SparePartsPrice de la Tabla Details, de decimal a int

También cambio en la Tabla Procedures el campo Price de decimal a int

Cambiamos en el modelo **histoy.dart**:

class History {

  int id = 0;

  String date = '';

  String dateLocal = '';

  int mileage = 0;

  String? remarks = '';

  List<Detail> details = [];

  int detailsCount = 0;

  int totalLabor = 0;

  int totalSpareParts = 0;

  int total = 0;

Cambiamos en el modelo **detail.dart**:

class Detail {

  int id = 0;

  Procedure procedure = Procedure(description: '', id: 0, price: 0);

  int laborPrice = 0;

  int sparePartsPrice = 0;

  int totalPrice = 0;

  String? remarks = '';

Cambiamos en el modelo **procedure.dart**:

class Procedure {

  var id = 0;

  String description = '';

  int price = 0;

Completamos la **history\_screen.dart**:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:connectivity/connectivity.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

class HistoryScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final History history;

  HistoryScreen(

      {required this.token,

      required this.user,

      required this.vehicle,

      required this.history});

  @override

  \_HistoryScreenState createState() => \_HistoryScreenState();

}

class \_HistoryScreenState extends State<HistoryScreen> {

  bool \_showLoader = false;

  String \_remarks = '';

  String \_remarksError = '';

  bool \_remarksShowError = false;

  TextEditingController \_remarksController = TextEditingController();

  String \_mileage = '';

  String \_mileageError = '';

  bool \_mileageShowError = false;

  TextEditingController \_mileageController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_remarks = widget.history.remarks!;

    \_remarksController.text = \_remarks;

    \_mileage = widget.history.mileage.toString();

    \_mileageController.text = \_mileage;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(

              widget.history.id == 0 ? 'Nueva Historia' : 'Editar Historia'),

        ),

        body: Stack(

          children: [

            Column(

              children: <Widget>[

                \_showRemarks(),

                \_showMileage(),

                \_showButtons(),

              ],

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showRemarks() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        keyboardType: TextInputType.multiline,

        minLines: 4,

        maxLines: 4,

        controller: \_remarksController,

        decoration: InputDecoration(

            hintText: 'Ingresa un Comentario...',

            labelText: 'Comentario',

            errorText: \_remarksShowError ? \_remarksError : null,

            suffixIcon: Icon(Icons.description),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_remarks = value;

        },

      ),

    );

  }

  Widget \_showMileage() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType: TextInputType.number,

        autofocus: true,

        controller: \_mileageController,

        decoration: InputDecoration(

            hintText: 'Ingrese Kilometraje...',

            labelText: 'Kilometraje',

            errorText: \_mileageShowError ? \_mileageError : null,

            suffixIcon: Icon(Icons.directions\_car),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_mileage = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.history.id == 0

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.history.id == 0

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

  void \_save() {

    if (!validateFields()) {

      return;

    }

    widget.history.id == 0 ? \_addRecord() : \_saveRecord();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_remarks.isEmpty) {

      isValid = false;

      \_remarksShowError = true;

      \_remarksError = 'Debes ingresar un Comentario';

    } else {

      \_remarksShowError = false;

    }

    if (\_mileage.isEmpty) {

      isValid = false;

      \_mileageShowError = true;

      \_mileageError = 'Debes ingresar un Kilometraje';

    } else {

      int mileage = int.parse(\_mileage);

      if (mileage <= 0) {

        isValid = false;

        \_mileageShowError = true;

        \_mileageError = 'Debes ingresar un Kilometraje mayor a cero';

      } else {

        \_mileageShowError = false;

      }

    }

    setState(() {});

    return isValid;

  }

  \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Map<String, dynamic> request = {

      'vehicleId': widget.vehicle.id,

      'mileage': int.parse(\_mileage),

      'remarks': \_remarks,

    };

    Response response =

        await ApiHelper.post('/api/Histories/', request, widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  \_saveRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Map<String, dynamic> request = {

      'id': widget.history.id,

      'vehicleId': widget.vehicle.id,

      'mileage': int.parse(\_mileage),

      'remarks': \_remarks,

    };

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.put(

        '/api/Histories/', widget.history.id.toString(), request, widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

  void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.delete(

        '/api/Histories/', widget.history.id.toString(), widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

}

# Detalle de una Historia

Creamos el archivo **detail\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/detail.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

class DetailScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final History history;

  final Detail detail;

  DetailScreen(

      {required this.token,

      required this.user,

      required this.vehicle,

      required this.history,

      required this.detail});

  @override

  \_DetailScreenState createState() => \_DetailScreenState();

}

class \_DetailScreenState extends State<DetailScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Detail'),

      ),

      body: Center(

        child: Text('Detail'),

      ),

    );

  }

}

Creamos el archivo **history\_info\_screen.dart**

import 'package:flutter/material.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

class HistoryInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final History history;

  HistoryInfoScreen(

      {required this.token,

      required this.user,

      required this.vehicle,

      required this.history,

      });

  @override

  \_HistoryInfoScreenState createState() => \_HistoryInfoScreenState();

}

class \_HistoryInfoScreenState extends State<HistoryInfoScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('History Info'),

      ),

      body: Center(

        child: Text('History Info'),

      ),

    );

  }

}

Modificamos en **vehicle\_info\_screen**

 @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

            '${\_vehicle.brand.description} ${\_vehicle.line} ${\_vehicle.plaque}'),

      ),

      body: Center(

        child: \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAddHistory(History(

            date: '',

            dateLocal: '',

            details: [],

            detailsCount: 0,

            id: 0,

            mileage: 0,

            remarks: '',

            total: 0,

            totalLabor: 0,

            totalSpareParts: 0)),

      ),

    );

  }

void \_goHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryInfoScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: \_vehicle,

                  history: history,

                )));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

  void \_goAddHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: \_vehicle,

                  history: history,

                )));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

Modificamos la **history\_info\_screen.dart**

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:cached\_network\_image/cached\_network\_image.dart';

import 'package:connectivity/connectivity.dart';

import 'package:flutter/material.dart';

import 'package:intl/intl.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/detail.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

import 'package:vehicles\_app/screens/detail\_screen.dart';

import 'package:vehicles\_app/screens/history\_screen.dart';

import 'package:vehicles\_app/screens/vehicle\_screen.dart';

class HistoryInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final History history;

  HistoryInfoScreen(

      {required this.token,

      required this.user,

      required this.vehicle,

      required this.history});

  @override

  \_HistoryInfoScreenState createState() => \_HistoryInfoScreenState();

}

class \_HistoryInfoScreenState extends State<HistoryInfoScreen> {

  bool \_showLoader = false;

  late History \_history;

  late Vehicle \_vehicle;

  @override

  void initState() {

    super.initState();

    \_history = widget.history;

    \_vehicle = widget.vehicle;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

            '${widget.vehicle.brand.description} ${widget.vehicle.line} ${widget.vehicle.plaque}'),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(

                text: 'Por favor espere...',

              )

            : \_getContent(),

      ),

      floatingActionButton: FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goDetail(Detail(

            id: 0,

            procedure: Procedure(id: 0, description: '', price: 0),

            laborPrice: 0,

            sparePartsPrice: 0,

            totalPrice: 0,

            remarks: '')),

      ),

    );

  }

  Widget \_getContent() {

    return Column(

      children: <Widget>[

        \_showVehicleInfo(),

        Expanded(

          child: \_history.details.length == 0 ? \_noContent() : \_getListView(),

        ),

      ],

    );

  }

  void \_goDetail(Detail detail) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => DetailScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: widget.vehicle,

                  history: widget.history,

                  detail: detail,

                )));

    if (result == 'yes') {

      await \_getHistory();

    }

  }

  Future<Null> \_getHistory() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response =

        await ApiHelper.getHistory(widget.token, widget.history.id.toString());

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_history = response.result;

    });

  }

  Widget \_showVehicleInfo() {

    return Container(

      margin: EdgeInsets.all(10),

      padding: EdgeInsets.all(5),

      child: Column(

        children: [

          Row(

            children: <Widget>[

              Stack(

                children: <Widget>[

                  ClipRRect(

                    borderRadius: BorderRadius.circular(5),

                    child: CachedNetworkImage(

                      imageUrl: \_vehicle.imageFullPath,

                      errorWidget: (context, url, error) => Icon(Icons.error),

                      fit: BoxFit.cover,

                      height: 100,

                      width: 100,

                      placeholder: (context, url) => Image(

                        image: AssetImage('assets/vehicles\_logo.png'),

                        fit: BoxFit.cover,

                        height: 100,

                        width: 100,

                      ),

                    ),

                  ),

                  Positioned(

                      bottom: 0,

                      left: 60,

                      child: InkWell(

                        onTap: () => \_goEditVehicle(),

                        child: ClipRRect(

                          borderRadius: BorderRadius.circular(30),

                          child: Container(

                            color: Colors.green[50],

                            height: 40,

                            width: 40,

                            child: Icon(

                              Icons.edit,

                              size: 30,

                              color: Colors.blue,

                            ),

                          ),

                        ),

                      ))

                ],

              ),

              Expanded(

                child: Stack(

                  children: [

                    Container(

                      margin: EdgeInsets.symmetric(horizontal: 10),

                      child: Row(

                        mainAxisAlignment: MainAxisAlignment.start,

                        children: [

                          Expanded(

                            child: Column(

                              mainAxisAlignment: MainAxisAlignment.start,

                              children: <Widget>[

                                Row(

                                  children: <Widget>[

                                    Text('Tipo de vehículo: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.vehicleType.description,

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('Marca: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.brand.description,

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('Modelo: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.model.toString(),

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('Placa: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.plaque,

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('Línea: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.line,

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('Color: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.color,

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('Comentarios: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.remarks == null

                                          ? 'NA'

                                          : widget.vehicle.remarks!,

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                                SizedBox(

                                  height: 5,

                                ),

                                Row(

                                  children: <Widget>[

                                    Text('N° de Historias: ',

                                        style: TextStyle(

                                            fontWeight: FontWeight.bold)),

                                    Text(

                                      \_vehicle.historiesCount.toString(),

                                      style: TextStyle(

                                        fontSize: 14,

                                      ),

                                    ),

                                  ],

                                ),

                              ],

                            ),

                          ),

                        ],

                      ),

                    ),

                  ],

                ),

              ),

            ],

          ),

          Stack(

            children: [

              Column(

                children: <Widget>[

                  SizedBox(

                    height: 5,

                  ),

                  Divider(

                    color: Colors.black,

                    height: 2,

                  ),

                  SizedBox(

                    height: 5,

                  ),

                  Row(

                    children: <Widget>[

                      Text('Descripción: ',

                          style: TextStyle(fontWeight: FontWeight.bold)),

                      Text(

                        \_history.remarks == null ? 'NA' : \_history.remarks!,

                        style: TextStyle(

                          fontSize: 14,

                        ),

                      ),

                    ],

                  ),

                  SizedBox(

                    height: 5,

                  ),

                  Row(

                    children: <Widget>[

                      Text('Kilometraje: ',

                          style: TextStyle(fontWeight: FontWeight.bold)),

                      Text(

                        \_history.mileage.toString(),

                        style: TextStyle(

                          fontSize: 14,

                        ),

                      ),

                    ],

                  ),

                  SizedBox(

                    height: 5,

                  ),

                  Row(

                    children: <Widget>[

                      Text('Valor Repuestos: ',

                          style: TextStyle(fontWeight: FontWeight.bold)),

                      Text(

                        '${NumberFormat.currency(symbol: '\$').format(\_history.totalSpareParts)}',

                        style: TextStyle(

                          fontSize: 14,

                        ),

                      ),

                    ],

                  ),

                  SizedBox(

                    height: 5,

                  ),

                  Row(

                    children: <Widget>[

                      Text('Valor Mano Obra: ',

                          style: TextStyle(fontWeight: FontWeight.bold)),

                      Text(

                        '${NumberFormat.currency(symbol: '\$').format(\_history.totalLabor)}',

                        style: TextStyle(

                          fontSize: 14,

                        ),

                      ),

                    ],

                  ),

                  SizedBox(

                    height: 5,

                  ),

                  Row(

                    children: <Widget>[

                      Text('Valor Total: ',

                          style: TextStyle(fontWeight: FontWeight.bold)),

                      Text(

                        '${NumberFormat.currency(symbol: '\$').format(\_history.total)}',

                        style: TextStyle(

                          fontSize: 14,

                        ),

                      ),

                    ],

                  ),

                ],

              ),

              Positioned(

                  bottom: 0,

                  left: 280,

                  child: InkWell(

                    onTap: () => \_goEditHistory(),

                    child: ClipRRect(

                      borderRadius: BorderRadius.circular(30),

                      child: Container(

                        color: Colors.green[50],

                        height: 40,

                        width: 40,

                        child: Icon(

                          Icons.edit,

                          size: 30,

                          color: Colors.blue,

                        ),

                      ),

                    ),

                  ))

            ],

          ),

        ],

      ),

    );

  }

  Widget \_getListView() {

    return ListView(

      children: \_history.details.map((e) {

        return Card(

          child: InkWell(

            onTap: () => \_goDetail(e),

            child: Container(

              margin: EdgeInsets.all(10),

              padding: EdgeInsets.all(5),

              child: Row(

                children: <Widget>[

                  Expanded(

                      child: Container(

                    margin: EdgeInsets.symmetric(horizontal: 10),

                    child: Row(

                      mainAxisAlignment: MainAxisAlignment.start,

                      children: <Widget>[

                        Column(

                          children: <Widget>[

                            Text(

                              e.procedure.description,

                              style: TextStyle(

                                  fontSize: 14, fontWeight: FontWeight.bold),

                            ),

                            Text(

                              e.remarks == null ? 'NA' : e.remarks!,

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                            Row(

                              children: [

                                Text(

                                  'Mano de obra: ${NumberFormat.currency(symbol: '\$').format(e.laborPrice)}',

                                  style: TextStyle(

                                    fontSize: 14,

                                  ),

                                ),

                              ],

                            ),

                            Row(

                              children: [

                                Text(

                                  'Repuestos: ${NumberFormat.currency(symbol: '\$').format(e.sparePartsPrice)}',

                                  style: TextStyle(

                                    fontSize: 14,

                                  ),

                                ),

                              ],

                            ),

                            Row(

                              children: [

                                Text(

                                  'Total: ${NumberFormat.currency(symbol: '\$').format(e.totalPrice)}',

                                  style: TextStyle(

                                    fontSize: 14,

                                  ),

                                ),

                              ],

                            ),

                          ],

                        ),

                      ],

                    ),

                  )),

                  Icon(

                    Icons.arrow\_forward\_ios,

                    size: 40,

                  )

                ],

              ),

            ),

          ),

        );

      }).toList(),

    );

  }

  Widget \_noContent() {

    return Center(

      child: Container(

        margin: EdgeInsets.all(20),

        child: Text(

          'La historia no tiene detalles registrados.',

          style: TextStyle(

            fontSize: 16,

            fontWeight: FontWeight.bold,

          ),

        ),

      ),

    );

  }

  void \_goEditHistory() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: widget.vehicle,

                  history: \_history,

                )));

    if (result == 'yes') {

      await \_getHistory();

    }

  }

  void \_goEditVehicle() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: widget.vehicle,

                )));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

  Future<Null> \_getVehicle() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response =

        await ApiHelper.getVehicle(widget.token, \_vehicle.id.toString());

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_vehicle = response.result;

    });

  }

}

En el Proyecto de Visual Studio en la carpeta Models/Request creamos la clase **DetailRequest**

using System.ComponentModel.DataAnnotations;

namespace Vehicles.API.Models.Request

{

public class DetailRequest

{

public int Id { get; set; }

[Display(Name = "Historia")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int HistoryId { get; set; }

[Display(Name = "Procedimiento")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int ProcedureId { get; set; }

[Display(Name = "Precio Mano de Obra")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int LaborPrice { get; set; }

[Display(Name = "Precio Repuestos")]

[DisplayFormat(DataFormatString = "{0:C2}")]

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

public int SparePartsPrice { get; set; }

[Display(Name = "Observación")]

[DataType(DataType.MultilineText)]

public string Remarks { get; set; }

}

}

Creamos en la carpeta **Controllers/API** el controlador **DetailsController**

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

using Vehicles.Api.Data;

using Vehicles.Api.Data.Entities;

using Vehicles.API.Models.Request;

namespace Vehicles.API.Controllers.API

{

[ApiController]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

[Route("api/[controller]")]

public class DetailsController : ControllerBase

{

private readonly DataContext \_context;

public DetailsController(DataContext context)

{

\_context = context;

}

[HttpPost]

public async Task<ActionResult<Detail>> PostDetail(DetailRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

History history = await \_context.Histories.FindAsync(request.HistoryId);

if (history == null)

{

return BadRequest("La historia no existe.");

}

Procedure procedure = await \_context.Procedures.FindAsync(request.ProcedureId);

if (procedure == null)

{

return BadRequest("El procedimiento no existe.");

}

Detail detail = new()

{

History = history,

LaborPrice = request.LaborPrice,

Procedure = procedure,

Remarks = request.Remarks,

SparePartsPrice = request.SparePartsPrice

};

\_context.Details.Add(detail);

await \_context.SaveChangesAsync();

return Ok(detail);

}

[HttpPut("{id}")]

public async Task<IActionResult> PutDetail(int id, DetailRequest request)

{

if (id != request.Id)

{

return BadRequest();

}

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

Procedure procedure = await \_context.Procedures.FindAsync(request.ProcedureId);

if (procedure == null)

{

return BadRequest("El procedimiento no existe.");

}

Detail detail = await \_context.Details.FindAsync(request.Id);

if (detail == null)

{

return BadRequest("El detalle no existe.");

}

detail.Procedure = procedure;

detail.Remarks = request.Remarks;

detail.SparePartsPrice = request.SparePartsPrice;

detail.LaborPrice = request.LaborPrice;

\_context.Details.Update(detail);

await \_context.SaveChangesAsync();

return NoContent();

}

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteDetail(int id)

{

Detail detail = await \_context.Details.FindAsync(id);

if (detail == null)

{

return NotFound();

}

\_context.Details.Remove(detail);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

Modificamos la **detail\_screen.dart**:

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:connectivity/connectivity.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/detail.dart';

import 'package:vehicles\_app/models/history.dart';

import 'package:vehicles\_app/models/procedure.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

import 'package:vehicles\_app/models/user.dart';

import 'package:vehicles\_app/models/vehicle.dart';

class DetailScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final History history;

  final Detail detail;

  DetailScreen(

      {required this.token,

      required this.user,

      required this.vehicle,

      required this.history,

      required this.detail});

  @override

  \_DetailScreenState createState() => \_DetailScreenState();

}

class \_DetailScreenState extends State<DetailScreen> {

  bool \_showLoader = false;

  int \_procedureId = 0;

  String \_procedureIdError = '';

  bool \_procedureIdShowError = false;

  List<Procedure> \_procedures = [];

  String \_remarks = '';

  String \_remarksError = '';

  bool \_remarksShowError = false;

  TextEditingController \_remarksController = TextEditingController();

  String \_laborPrice = '';

  String \_laborPriceError = '';

  bool \_laborPriceShowError = false;

  TextEditingController \_laborPriceController = TextEditingController();

  String \_sparePartsPrice = '';

  String \_sparePartsPriceError = '';

  bool \_sparePartsPriceShowError = false;

  TextEditingController \_sparePartsPriceController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_getProcedures();

    \_loadFieldValues();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text(widget.detail.id == 0

              ? 'Nuevo Procedimiento'

              : widget.detail.procedure.description),

        ),

        body: Stack(

          children: [

            SingleChildScrollView(

              child: Column(

                children: <Widget>[

                  \_showProcedure(),

                  \_showRemarks(),

                  \_showLaborPrices(),

                  \_showSparePartsPrice(),

                  \_showButtons(),

                ],

              ),

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Future<Null> \_getProcedures() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.getProcedures(widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_procedures = response.result;

    });

  }

  Widget \_showProcedure() {

    return Container(

      padding: EdgeInsets.all(10),

      child: \_procedures.length == 0

          ? Text('Cargando procedimientos...')

          : DropdownButtonFormField(

              items: \_getComboProcedures(),

              value: \_procedureId,

              onChanged: (option) {

                setState(() {

                  \_procedureId = option as int;

                });

              },

              decoration: InputDecoration(

                hintText: 'Seleccione un Procedimiento...',

                labelText: 'Procedimiento',

                errorText: \_procedureIdShowError ? \_procedureIdError : null,

                border:

                    OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

              )),

    );

  }

  Widget \_showRemarks() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType: TextInputType.multiline,

        minLines: 4,

        maxLines: 4,

        controller: \_remarksController,

        decoration: InputDecoration(

            hintText: 'Ingresa un Comentario...',

            labelText: 'Comentario',

            errorText: \_remarksShowError ? \_remarksError : null,

            suffixIcon: Icon(Icons.description),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_remarks = value;

        },

      ),

    );

  }

  Widget \_showLaborPrices() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType:

            TextInputType.numberWithOptions(decimal: true, signed: false),

        controller: \_laborPriceController,

        decoration: InputDecoration(

            hintText: 'Ingresa un Precio de Mano de Obra...',

            labelText: 'Precio de Mano de Obra',

            errorText: \_laborPriceShowError ? \_laborPriceError : null,

            suffixIcon: Icon(Icons.build),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_laborPrice = value;

        },

      ),

    );

  }

  Widget \_showSparePartsPrice() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        keyboardType:

            TextInputType.numberWithOptions(decimal: true, signed: false),

        controller: \_sparePartsPriceController,

        decoration: InputDecoration(

            hintText: 'Ingresa un Precio de Respuestos...',

            labelText: 'Precio de Respuestos',

            errorText: \_sparePartsPriceShowError ? \_sparePartsPriceError : null,

            suffixIcon: Icon(Icons.attach\_money),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_sparePartsPrice = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.detail.id == 0

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.detail.id == 0

              ? Container()

              : Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

  List<DropdownMenuItem<int>> \_getComboProcedures() {

    List<DropdownMenuItem<int>> list = [];

    list.add(DropdownMenuItem(

      child: Text('Seleccione un Procedimiento...'),

      value: 0,

    ));

    \_procedures.forEach((procedure) {

      list.add(DropdownMenuItem(

        child: Text(procedure.description),

        value: procedure.id,

      ));

    });

    return list;

  }

  \_save() {}

  \_confirmDelete() {}

  void \_loadFieldValues() {

    \_procedureId = widget.detail.procedure.id;

    \_remarks = widget.detail.remarks == null ? 'N/A' : widget.detail.remarks!;

    // \_remarks = widget.detail.remarks!;

    \_remarksController.text = \_remarks;

    \_laborPrice = widget.detail.laborPrice.toString();

    \_laborPriceController.text = \_laborPrice;

    \_sparePartsPrice = widget.detail.sparePartsPrice.toString();

    \_sparePartsPriceController.text = \_sparePartsPrice;

  }

}

Agregamos los métodos **\_saveRecord**, **\_addRecord**, **\_confirmDelete** y **\_deleteRecord**

void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.delete(

        '/api/Brands/', widget.brand.id.toString(), widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

void \_confirmDelete() async {

    var response = await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: '¿Estás seguro de querer borrar el registro?',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: 'no', label: 'No'),

          AlertDialogAction(key: 'yes', label: 'Sí'),

        ]);

    if (response == 'yes') {

      \_deleteRecord();

    }

  }

  void \_deleteRecord() async {

    setState(() {

      \_showLoader = true;

    });

    Response response = await ApiHelper.delete(

        '/api/Details/', widget.detail.id.toString(), widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Navigator.pop(context, 'yes');

  }

Para que al elegir un procedimiento traiga el precio de mano de obra precargado hacemos:

Widget \_showProcedure() {

    return Container(

      padding: EdgeInsets.all(10),

      child: \_procedures.length == 0

          ? Text('Cargando procedimientos...')

          : DropdownButtonFormField(

              items: \_getComboProcedures(),

              value: \_procedureId,

              onChanged: (option) {

                setState(() {

                  \_procedureId = option as int;

                  \_laborPrice = \_getPrice(\_procedureId).toString();

                  \_laborPriceController.text = \_laborPrice;

                });

              },

              decoration: InputDecoration(

                hintText: 'Seleccione un Procedimiento...',

                labelText: 'Procedimiento',

                errorText: \_procedureIdShowError ? \_procedureIdError : null,

                border:

                    OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

              )),

    );

  }

 int \_getPrice(int procedureId) {

    var procedures = \_procedures.where((p) => p.id == procedureId).toList();

    return procedures[0].price;

  }

# Registrar Nuevo Usuario

Vamos al Proyecto de VisualStudio y en el **DocumentTypesController** del API hacemos:

// GET: api/DocumentTypes

[AllowAnonymous]

[HttpGet]

public async Task<ActionResult<IEnumerable<DocumentType>>> GetDocumentTypes()

{

return await \_context.DocumentTypes.OrderBy(x => x.Description).ToListAsync();

}

En la carpeta **Models/Request** creamos la Clase **RegisterRequest**:

using System.ComponentModel.DataAnnotations;

namespace Vehicles.API.Models.Request

{

public class RegisterRequest : UserRequest

{

[Required(ErrorMessage = "El campo {0} es obligatorio.")]

[MinLength(6, ErrorMessage = "El campo {0} debe tener una longitud mínima de {1} carácteres.")]

public string Password { get; set; }

}

}

En el **AccountController** del API creamos el método **PostUser**:

[HttpPost]

public async Task<ActionResult<User>> PostUser(RegisterRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

DocumentType documentType = await \_context.DocumentTypes.FindAsync(request.DocumentTypeId);

if (documentType == null)

{

return BadRequest("El tipo de documento no existe.");

}

User user = await \_userHelper.GetUserAsync(request.Email);

if (user != null)

{

return BadRequest("Ya existe un usuario registrado con ese email.");

}

string imageId = string.Empty;

if (request.Image != null && request.Image.Length > 0)

{

imageId = \_imageHelper.UploadImage(request.Image, "users");

}

user = new User

{

Address = request.Address,

Document = request.Document,

DocumentType = documentType,

Email = request.Email,

FirstName = request.FirstName,

ImageId = imageId,

LastName = request.LastName,

PhoneNumber = request.PhoneNumber,

UserName = request.Email,

UserType = UserType.User,

};

await \_userHelper.AddUserAsync(user, request.Password);

await \_userHelper.AddUserToRoleAsync(user, user.UserType.ToString());

string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

string tokenLink = Url.Action("ConfirmEmail", "Account", new

{

userid = user.Id,

token = myToken

}, protocol: HttpContext.Request.Scheme);

\_mailHelper.SendMail(user.Email, "Vehicles - Confirmación de cuenta", $"<h1>Vehicles - Confirmación de cuenta</h1>" +

$"Para habilitar el usuario, " +

$"por favor hacer clic en el siguiente enlace: </br></br><a href = \"{tokenLink}\">Confirmar Email</a>");

return Ok(user);

}

En el Proyecto Flutter de la App, modificamos en **api\_helper.dart** el método **getDocumentTypes** (le sacamos lo del token):

 static Future<Response> getDocumentTypes() async {

    var url = Uri.parse('${Constants.apiUrl}/api/DocumentTypes');

    var response = await http.get(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

      },

    );

    var body = response.body;

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: body);

    }

En **user\_screen.dart** y en **document\_types\_screen.dart** corregimos (sacamos el token) cuando se llama al método **getDocumentTypes**:

Response response = await ApiHelper.getDocumentTypes();

Agregamos al **api\_helper.dart** el método **postNoToken**:

static Future<Response> postNoToken(

      String controller, Map<String, dynamic> request) async {

    var url = Uri.parse('${Constants.apiUrl}$controller');

    var response = await http.post(

      url,

      headers: {

        'content-type': 'application/json',

        'accept': 'application/json',

      },

      body: jsonEncode(request),

    );

    if (response.statusCode >= 400) {

      return Response(isSuccess: false, message: response.body);

    }

    return Response(isSuccess: true);

  }

En la carpeta **screens** creamos el archivo **register\_user\_screen.dart**

import 'package:flutter/material.dart';

class RegisterUserScreen extends StatefulWidget {

  const RegisterUserScreen({Key? key}) : super(key: key);

  @override

  \_RegisterUserScreenState createState() => \_RegisterUserScreenState();

}

class \_RegisterUserScreenState extends State<RegisterUserScreen> {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Register new user'),

      ),

      body: Center(

        child: Text('Register New User'),

      ),

    );

  }

}

En la **login\_screen** hacemos:

Widget \_showRegisterButton() {

    return Expanded(

      child: ElevatedButton(

        child: Text('Nuevo Usuario'),

        style: ButtonStyle(

          backgroundColor: MaterialStateProperty.resolveWith<Color>(

            (Set<MaterialState> states) {

              return Color(0xFFE03B8B);

            }

          ),

        ),

        onPressed: () => \_register(),

      ),

    );

  }

void \_register() {

    Navigator.push(

        context, MaterialPageRoute(builder: (context) => RegisterUserScreen()));

  }

Y completamos la **register\_user\_screen.dart**

import 'dart:convert';

import 'dart:io';

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:cached\_network\_image/cached\_network\_image.dart';

import 'package:camera/camera.dart';

import 'package:connectivity/connectivity.dart';

import 'package:email\_validator/email\_validator.dart';

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/document\_type.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/screens/take\_picture\_screen.dart';

class RegisterUserScreen extends StatefulWidget {

  const RegisterUserScreen({Key? key}) : super(key: key);

  @override

  \_RegisterUserScreenState createState() => \_RegisterUserScreenState();

}

class \_RegisterUserScreenState extends State<RegisterUserScreen> {

  bool \_showLoader = false;

  bool \_photoChanged = false;

  late XFile \_image;

  String \_firstName = '';

  String \_firstNameError = '';

  bool \_firstNameShowError = false;

  TextEditingController \_firstNameController = TextEditingController();

  String \_lastName = '';

  String \_lastNameError = '';

  bool \_lastNameShowError = false;

  TextEditingController \_lastNameController = TextEditingController();

  int \_documentTypeId = 0;

  String \_documentTypeIdError = '';

  bool \_documentTypeIdShowError = false;

  List<DocumentType> \_documentTypes = [];

  String \_document = '';

  String \_documentError = '';

  bool \_documentShowError = false;

  TextEditingController \_documentController = TextEditingController();

  String \_address = '';

  String \_addressError = '';

  bool \_addressShowError = false;

  TextEditingController \_addressController = TextEditingController();

  String \_email = '';

  String \_emailError = '';

  bool \_emailShowError = false;

  TextEditingController \_emailController = TextEditingController();

  String \_phoneNumber = '';

  String \_phoneNumberError = '';

  bool \_phoneNumberShowError = false;

  TextEditingController \_phoneNumberController = TextEditingController();

  bool \_passwordShow = false;

  String \_password = '';

  String \_passwordError = '';

  bool \_passwordShowError = false;

  TextEditingController \_passwordController = TextEditingController();

  String \_confirm = '';

  String \_confirmError = '';

  bool \_confirmShowError = false;

  TextEditingController \_confirmController = TextEditingController();

  @override

  void initState() {

    super.initState();

    \_getDocumentTypes();

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text('Nuevo Usuario'),

        ),

        body: Stack(

          children: [

            SingleChildScrollView(

              child: Column(

                children: <Widget>[

                  \_showPhoto(),

                  \_showFirstName(),

                  \_showLastName(),

                  \_showDocumentType(),

                  \_showDocument(),

                  \_showAddress(),

                  \_showEmail(),

                  \_showPhoneNumber(),

                  \_showPassword(),

                  \_showConfirm(),

                  \_showButtons(),

                ],

              ),

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Future<Null> \_getDocumentTypes() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Response response = await ApiHelper.getDocumentTypes();

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    setState(() {

      \_documentTypes = response.result;

    });

  }

  List<DropdownMenuItem<int>> \_getComboDocumentTypes() {

    List<DropdownMenuItem<int>> list = [];

    list.add(DropdownMenuItem(

      child: Text('Seleccione un Tipo de Documento...'),

      value: 0,

    ));

    \_documentTypes.forEach((documentType) {

      list.add(DropdownMenuItem(

        child: Text(documentType.description),

        value: documentType.id,

      ));

    });

    return list;

  }

  Widget \_showPhoto() {

    return InkWell(

      child: Stack(children: <Widget>[

        Container(

          margin: EdgeInsets.only(top: 10),

          child: !\_photoChanged

              ? Image(

                  image: AssetImage('assets/noimage.png'),

                  width: 160,

                  height: 160,

                  fit: BoxFit.cover)

              : ClipRRect(

                  borderRadius: BorderRadius.circular(80),

                  child: Image.file(

                    File(\_image.path),

                    width: 160,

                    height: 160,

                    fit: BoxFit.cover,

                  )),

        ),

        Positioned(

            bottom: 0,

            left: 100,

            child: InkWell(

              onTap: () => \_takePicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.photo\_camera,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

        Positioned(

            bottom: 0,

            left: 0,

            child: InkWell(

              onTap: () => \_selectPicture(),

              child: ClipRRect(

                borderRadius: BorderRadius.circular(30),

                child: Container(

                  color: Colors.green[50],

                  height: 60,

                  width: 60,

                  child: Icon(

                    Icons.image,

                    size: 40,

                    color: Colors.blue,

                  ),

                ),

              ),

            )),

      ]),

    );

  }

  Widget \_showFirstName() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        controller: \_firstNameController,

        decoration: InputDecoration(

            hintText: 'Ingresa nombres...',

            labelText: 'Nombres',

            errorText: \_firstNameShowError ? \_firstNameError : null,

            suffixIcon: Icon(Icons.person),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_firstName = value;

        },

      ),

    );

  }

  Widget \_showLastName() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_lastNameController,

        decoration: InputDecoration(

            hintText: 'Ingresa nombres...',

            labelText: 'Apellido',

            errorText: \_lastNameShowError ? \_lastNameError : null,

            suffixIcon: Icon(Icons.person),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_lastName = value;

        },

      ),

    );

  }

  Widget \_showDocumentType() {

    return Container(

      padding: EdgeInsets.all(10),

      child: \_documentTypes.length == 0

          ? Text('Cargando tipos de documento...')

          : DropdownButtonFormField(

              items: \_getComboDocumentTypes(),

              value: \_documentTypeId,

              onChanged: (option) {

                setState(() {

                  \_documentTypeId = option as int;

                });

              },

              decoration: InputDecoration(

                hintText: 'Seleccione un tipo de documento...',

                labelText: 'Tipo de Documento',

                errorText:

                    \_documentTypeIdShowError ? \_documentTypeIdError : null,

                border:

                    OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

              )),

    );

  }

  Widget \_showDocument() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_documentController,

        decoration: InputDecoration(

            hintText: 'Ingresa documento...',

            labelText: 'Documento',

            errorText: \_documentShowError ? \_documentError : null,

            suffixIcon: Icon(Icons.assignment\_ind),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_document = value;

        },

      ),

    );

  }

  Widget \_showAddress() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_addressController,

        keyboardType: TextInputType.streetAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa dirección...',

            labelText: 'Dirección',

            errorText: \_addressShowError ? \_addressError : null,

            suffixIcon: Icon(Icons.home),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_address = value;

        },

      ),

    );

  }

  Widget \_showEmail() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_emailController,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

            hintText: 'Ingresa Email...',

            labelText: 'Email',

            errorText: \_emailShowError ? \_emailError : null,

            suffixIcon: Icon(Icons.email),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_email = value;

        },

      ),

    );

  }

  Widget \_showPhoneNumber() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        controller: \_phoneNumberController,

        keyboardType: TextInputType.phone,

        decoration: InputDecoration(

            hintText: 'Ingresa Teléfono...',

            labelText: 'Teléfono',

            errorText: \_phoneNumberShowError ? \_phoneNumberError : null,

            suffixIcon: Icon(Icons.phone),

            border:

                OutlineInputBorder(borderRadius: BorderRadius.circular(10))),

        onChanged: (value) {

          \_phoneNumber = value;

        },

      ),

    );

  }

  Widget \_showPassword() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: !\_passwordShow,

        decoration: InputDecoration(

          hintText: 'Ingresa una contraseña...',

          labelText: 'Contraseña',

          errorText: \_passwordShowError ? \_passwordError : null,

          prefixIcon: Icon(Icons.lock),

          suffixIcon: IconButton(

            icon: \_passwordShow

                ? Icon(Icons.visibility)

                : Icon(Icons.visibility\_off),

            onPressed: () {

              setState(() {

                \_passwordShow = !\_passwordShow;

              });

            },

          ),

          border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

        ),

        onChanged: (value) {

          \_password = value;

        },

      ),

    );

  }

  Widget \_showConfirm() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: !\_passwordShow,

        decoration: InputDecoration(

          hintText: 'Ingresa la confirmación de contraseña...',

          labelText: 'Confirmación de contraseña',

          errorText: \_confirmShowError ? \_confirmError : null,

          prefixIcon: Icon(Icons.lock),

          suffixIcon: IconButton(

            icon: \_passwordShow

                ? Icon(Icons.visibility)

                : Icon(Icons.visibility\_off),

            onPressed: () {

              setState(() {

                \_passwordShow = !\_passwordShow;

              });

            },

          ),

          border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

        ),

        onChanged: (value) {

          \_confirm = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          \_showRegisterButton(),

        ],

      ),

    );

  }

  void \_takePicture() async {

    WidgetsFlutterBinding.ensureInitialized();

    final cameras = await availableCameras();

    final firstCamera = cameras.first;

    Response? response = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => TakePictureScreen(

                  camera: firstCamera,

                )));

    if (response != null) {

      setState(() {

        \_photoChanged = true;

        \_image = response.result;

      });

    }

  }

  void \_selectPicture() async {

    final ImagePicker \_picker = ImagePicker();

    final XFile? image = await \_picker.pickImage(source: ImageSource.gallery);

    if (image != null) {

      setState(() {

        \_photoChanged = true;

        \_image = image;

      });

    }

  }

  Widget \_showRegisterButton() {

    return Expanded(

      child: ElevatedButton(

        child: Text('Registrar Usuario'),

        style: ButtonStyle(

          backgroundColor: MaterialStateProperty.resolveWith<Color>(

              (Set<MaterialState> states) {

            return Color(0xFF120E43);

          }),

        ),

        onPressed: () => \_register(),

      ),

    );

  }

  void \_register() async {

    if (!validateFields()) {

      return;

    }

    \_addRecord();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_firstName.isEmpty) {

      isValid = false;

      \_firstNameShowError = true;

      \_firstNameError = 'Debes ingresar un nombre';

    } else {

      \_firstNameShowError = false;

    }

    if (\_lastName.isEmpty) {

      isValid = false;

      \_lastNameShowError = true;

      \_lastNameError = 'Debes ingresar un apellido';

    } else {

      \_lastNameShowError = false;

    }

    if (\_documentTypeId == 0) {

      isValid = false;

      \_documentTypeIdShowError = true;

      \_documentTypeIdError = 'Debes seleccionar un Tipo de Documento';

    } else {

      \_documentTypeIdShowError = false;

    }

    if (\_document.isEmpty) {

      isValid = false;

      \_documentShowError = true;

      \_documentError = 'Debes ingresar un documento';

    } else {

      \_documentShowError = false;

    }

    if (\_address.isEmpty) {

      isValid = false;

      \_addressShowError = true;

      \_addressError = 'Debes ingresar una dirección';

    } else {

      \_addressShowError = false;

    }

    if (\_email.isEmpty) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar tu Email';

    } else if (!EmailValidator.validate(\_email)) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar un Email válido';

    } else {

      \_emailShowError = false;

    }

    if (\_phoneNumber.isEmpty) {

      isValid = false;

      \_phoneNumberShowError = true;

      \_phoneNumberError = 'Debes ingresar un teléfono';

    } else {

      \_phoneNumberShowError = false;

    }

    if (\_password.length < 6) {

      isValid = false;

      \_passwordShowError = true;

      \_passwordError = 'Debes ingresar una Contraseña de al menos 6 caracteres';

    } else {

      \_phoneNumberShowError = false;

    }

    if (\_confirm.length < 6) {

      isValid = false;

      \_confirmShowError = true;

      \_confirmError =

          'Debes ingresar una Confirmación de Contraseña de al menos 6 caracteres';

    } else {

      \_phoneNumberShowError = false;

    }

    if (\_confirm != \_password) {

      isValid = false;

      \_passwordShowError = true;

      \_confirmShowError = true;

      \_passwordError = 'La contraseña y la confirmación no son iguales';

      \_confirmError = 'La contraseña y la confirmación no son iguales';

    } else {

      \_passwordShowError = false;

      \_confirmShowError = false;

    }

    setState(() {});

    return isValid;

  }

  void \_addRecord() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estés conectado a Internet',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    String base64image = '';

    if (\_photoChanged) {

      List<int> imageBytes = await \_image.readAsBytes();

      base64image = base64Encode(imageBytes);

    }

    Map<String, dynamic> request = {

      'firstName': \_firstName,

      'lastName': \_lastName,

      'documentTypeId': \_documentTypeId,

      'document': \_document,

      'address': \_address,

      'email': \_email,

      'userName': \_email,

      'phoneNumber': \_phoneNumber,

      'password': \_password,

      'image': base64image,

    };

    Response response = await ApiHelper.postNoToken('/api/Account/', request);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message:

            'Se ha enviado un correo con las instrucciones para activar el usuario. Por favor actívelo para poder ingresar a la Aplicación.',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: null, label: 'Aceptar'),

        ]);

    return;

    Navigator.pop(context, 'yes');

  }

}

# Editar Usuario

En **user\_screen.dart** hacemos:

class UserScreen extends StatefulWidget {

  final Token token;

  final User user;

  final bool myProfile;

  UserScreen({required this.token, required this.user, required this.myProfile});

En **user\_info\_screen.dart** hacemos:

void \_goEdit() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => UserScreen(

                  token: widget.token,

                  user: \_user,

                  myProfile: false,

                )));

    if (result == 'yes') {

      //TODO: Pending refresh user info

    }

  }

En **user\_screen.dart** hacemos:

void \_goAdd() async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => UserScreen(

                token: widget.token,

                user: User(

                  firstName: '',

                  lastName: '',

                  documentType: DocumentType(id: 0, description: ''),

                  document: '',

                  address: '',

                  imageId: '',

                  imageFullPath: '',

                  userType: 1,

                  fullName: '',

                  vehicles: [],

                  vehiclesCount: 0,

                  id: '',

                  userName: '',

                  email: '',

                  phoneNumber: '',

                ),

                myProfile: false,)));

    if (result == 'yes') {

      \_getUsers();

    }

  }

En **user\_screen.dart** hacemos:

Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          Expanded(

            child: ElevatedButton(

              child: Text('Guardar'),

              style: ButtonStyle(

                backgroundColor: MaterialStateProperty.resolveWith<Color>(

                    (Set<MaterialState> states) {

                  return Color(0xFF120E43);

                }),

              ),

              onPressed: () => \_save(),

            ),

          ),

          widget.user.id.isEmpty

              ? Container()

              : SizedBox(

                  width: 20,

                ),

          widget.user.id.isEmpty

              ? Container()

              :widget.myProfile ? Expanded(

                  child: ElevatedButton(

                    child: Text('Cambiar Contraseña'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_changePassword(),

                  ),

                ): Expanded(

                  child: ElevatedButton(

                    child: Text('Borrar'),

                    style: ButtonStyle(

                      backgroundColor: MaterialStateProperty.resolveWith<Color>(

                          (Set<MaterialState> states) {

                        return Color(0xFFB4161B);

                      }),

                    ),

                    onPressed: () => \_confirmDelete(),

                  ),

                ),

        ],

      ),

    );

  }

En **home\_screen.dart** hacemos:

ListTile(

            leading: Icon(Icons.face),

            title: Text('Editar perfil'),

             onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => User2Screen(

                            token: widget.token,user: widget.token.user2 ,myProfile: true,

                          )));

            },

          ),

tanto en el menú del Mecánico como en el menú del Cliente

En el Proyecto de Visual Studio, en el Controlador API **UsersController** cambiamos en el método **PutUser**:

// PUT: api/Users/5

[HttpPut("{id}")]

public async Task<IActionResult> PutUser(string id, UserRequest request)

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

DocumentType documentType = await \_context.DocumentTypes.FindAsync(request.DocumentTypeId);

if (documentType == null)

{

return BadRequest("El tipo de documento no existe.");

}

User user = await \_userHelper.GetUserAsync(request.Email);

if (user == null)

{

return BadRequest("No existe el usuario.");

}

string imageId = user.ImageId;

if (request.Image != null && request.Image.Length > 0)

{

imageId = \_imageHelper.UploadImage(request.Image, "users");

}

user.Address = request.Address;

user.Document = request.Document;

user.DocumentType = documentType;

user.FirstName = request.FirstName;

user.ImageId = imageId;

user.LastName = request.LastName;

user.PhoneNumber = request.PhoneNumber;

await \_userHelper.UpdateUserAsync(user);

return Ok(user);

}

En el Proyecto de Visual Studio, en el Controlador API **AccountController** creamos los métodos **ChangePassword** y **RecoverPassword**

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

[HttpPost]

[Route("ChangePassword")]

public async Task<IActionResult> ChangePassword(ChangePasswordViewModel model)

{

if (ModelState.IsValid)

{

string email = User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.NameIdentifier).Value;

User user = await \_userHelper.GetUserAsync(email);

if (user != null)

{

IdentityResult result = await \_userHelper.ChangePasswordAsync(user, model.OldPassword, model.NewPassword);

if (result.Succeeded)

{

return NoContent();

}

else

{

return BadRequest(result.Errors.FirstOrDefault().Description);

}

}

else

{

return BadRequest("Usuario no encontrado.");

}

}

return BadRequest(ModelState);

}

[HttpPost]

[Route("RecoverPassword")]

public async Task<IActionResult> RecoverPassword(RecoverPasswordViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userHelper.GetUserAsync(model.Email);

if (user == null)

{

return BadRequest("El correo ingresado no corresponde a ningún usuario.");

}

string myToken = await \_userHelper.GeneratePasswordResetTokenAsync(user);

string link = Url.Action(

"ResetPassword",

"Account",

new { token = myToken }, protocol: HttpContext.Request.Scheme);

\_mailHelper.SendMail(model.Email, "Vehicles - Reseteo de contraseña", $"<h1>Vehicles - Reseteo de contraseña</h1>" +

$"Para establecer una nueva contraseña haga clic en el siguiente enlace:</br></br>" +

$"<a href = \"{link}\">Cambio de Contraseña</a>");

return Ok("Las instrucciones para el cambio de contraseña han sido enviadas a su email.");

}

return BadRequest(model);

}

# Cambiar Contraseña

En **user\_screen.dart** hacemos:

 void \_changePassword() {

    Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => ChangePasswordScreen(

                  token: widget.token,

                )));

  }

Creamos el archivo **change\_password\_screen.dart**

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:connectivity/connectivity.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/response.dart';

import 'package:vehicles\_app/models/token.dart';

class ChangePasswordScreen extends StatefulWidget {

  final Token token;

  ChangePasswordScreen({required this.token});

  @override

  \_ChangePasswordScreenState createState() => \_ChangePasswordScreenState();

}

class \_ChangePasswordScreenState extends State<ChangePasswordScreen> {

  bool \_showLoader = false;

  String \_currentPassword = '';

  String \_currentPasswordError = '';

  bool \_currentPasswordShowError = false;

  TextEditingController \_currentPasswordController = TextEditingController();

  String \_newPassword = '';

  String \_newPasswordError = '';

  bool \_newPasswordShowError = false;

  TextEditingController \_newPasswordController = TextEditingController();

  String \_confirmPassword = '';

  String \_confirmPasswordError = '';

  bool \_confirmPasswordShowError = false;

  TextEditingController \_confirmPasswordController = TextEditingController();

  bool \_passwordShow = false;

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        appBar: AppBar(

          title: Text('Cambio de Contraseña'),

        ),

        body: Stack(

          children: [

            Column(

              children: <Widget>[

                \_showCurrentPassword(),

                \_showNewPassword(),

                \_showConfirmPassword(),

                \_showButtons(),

              ],

            ),

            \_showLoader

                ? LoaderComponent(

                    text: 'Por favor espere...',

                  )

                : Container(),

          ],

        ));

  }

  Widget \_showCurrentPassword() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: !\_passwordShow,

        decoration: InputDecoration(

          hintText: 'Ingresa la contraseña actual...',

          labelText: 'Contraseña actual',

          errorText: \_currentPasswordShowError ? \_currentPasswordError : null,

          prefixIcon: Icon(Icons.lock),

          suffixIcon: IconButton(

            icon: \_passwordShow

                ? Icon(Icons.visibility)

                : Icon(Icons.visibility\_off),

            onPressed: () {

              setState(() {

                \_passwordShow = !\_passwordShow;

              });

            },

          ),

          border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

        ),

        onChanged: (value) {

          \_currentPassword = value;

        },

      ),

    );

  }

  Widget \_showNewPassword() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: !\_passwordShow,

        decoration: InputDecoration(

          hintText: 'Ingresa la nueva contraseña...',

          labelText: 'Nueva Contraseña',

          errorText: \_newPasswordShowError ? \_newPasswordError : null,

          prefixIcon: Icon(Icons.lock),

          suffixIcon: IconButton(

            icon: \_passwordShow

                ? Icon(Icons.visibility)

                : Icon(Icons.visibility\_off),

            onPressed: () {

              setState(() {

                \_passwordShow = !\_passwordShow;

              });

            },

          ),

          border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

        ),

        onChanged: (value) {

          \_newPassword = value;

        },

      ),

    );

  }

  Widget \_showConfirmPassword() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        obscureText: !\_passwordShow,

        decoration: InputDecoration(

          hintText: 'Confirmación de contraseña...',

          labelText: 'Confirmación de contraseña',

          errorText: \_confirmPasswordShowError ? \_confirmPasswordError : null,

          prefixIcon: Icon(Icons.lock),

          suffixIcon: IconButton(

            icon: \_passwordShow

                ? Icon(Icons.visibility)

                : Icon(Icons.visibility\_off),

            onPressed: () {

              setState(() {

                \_passwordShow = !\_passwordShow;

              });

            },

          ),

          border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

        ),

        onChanged: (value) {

          \_confirmPassword = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          \_showChangePassword(),

        ],

      ),

    );

  }

  Widget \_showChangePassword() {

    return Expanded(

      child: ElevatedButton(

        child: Text('Cambiar Contraseña'),

        style: ButtonStyle(

          backgroundColor: MaterialStateProperty.resolveWith<Color>(

              (Set<MaterialState> states) {

            return Color(0xFF120E43);

          }),

        ),

        onPressed: () => \_save(),

      ),

    );

  }

  void \_save() async {

    if (!validateFields()) {

      return;

    }

    \_changePassword();

  }

  bool validateFields() {

    bool isValid = true;

    if (\_currentPassword.length < 6) {

      isValid = false;

      \_currentPasswordShowError = true;

      \_currentPasswordError =

          'Debes ingresar tu Contraseña actual de al menos 6 caracteres';

    } else {

      \_currentPasswordShowError = false;

    }

    if (\_newPassword.length < 6) {

      isValid = false;

      \_newPasswordShowError = true;

      \_newPasswordError =

          'Debes ingresar una Contraseña de al menos 6 caracteres';

    } else {

      \_newPasswordShowError = false;

    }

    if (\_confirmPassword.length < 6) {

      isValid = false;

      \_confirmPasswordShowError = true;

      \_confirmPasswordError =

          'Debes ingresar una Confirmación de Contraseña de al menos 6 caracteres';

    } else {

      \_confirmPasswordShowError = false;

    }

    if (\_confirmPassword != \_newPassword) {

      isValid = false;

      \_newPasswordShowError = true;

      \_confirmPasswordShowError = true;

      \_newPasswordError = 'La contraseña y la confirmación no son iguales';

      \_confirmPasswordError = 'La contraseña y la confirmación no son iguales';

    } else {

      \_newPasswordShowError = false;

      \_confirmPasswordShowError = false;

    }

    setState(() {});

    return isValid;

  }

  void \_changePassword() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Map<String, dynamic> request = {

      'oldPassword': \_currentPassword,

      'newPassword': \_newPassword,

      'confirm': \_confirmPassword,

    };

    Response response = await ApiHelper.post(

        '/api/Account/ChangePassword', request, widget.token);

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message: 'Su contraseña ha sido cambiada con éxito.',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: null, label: 'Aceptar'),

        ]);

    Navigator.pop(context, 'yes');

  }

}

# Recuperar Contraseña

Creamos el archivo **recover\_password\_screen.dart**

import 'package:adaptive\_dialog/adaptive\_dialog.dart';

import 'package:connectivity/connectivity.dart';

import 'package:email\_validator/email\_validator.dart';

import 'package:flutter/material.dart';

import 'package:vehicles\_app/components/loader\_component.dart';

import 'package:vehicles\_app/helpers/api\_helper.dart';

import 'package:vehicles\_app/models/response.dart';

class RecoverPasswordScreen extends StatefulWidget {

  const RecoverPasswordScreen({Key? key}) : super(key: key);

  @override

  \_RecoverPasswordScreenState createState() => \_RecoverPasswordScreenState();

}

class \_RecoverPasswordScreenState extends State<RecoverPasswordScreen> {

  bool \_showLoader = false;

  String \_email = '';

  String \_emailError = '';

  bool \_emailShowError = false;

  TextEditingController \_emailController = TextEditingController();

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Recuperar contraseña'),

      ),

      body: Stack(

        children: <Widget>[

          Column(

            children: <Widget>[

              \_showEmail(),

              \_showButtons(),

            ],

          ),

          \_showLoader

              ? LoaderComponent(

                  text: 'Por favor espere...',

                )

              : Container(),

        ],

      ),

    );

  }

  Widget \_showEmail() {

    return Container(

      padding: EdgeInsets.all(10),

      child: TextField(

        autofocus: true,

        keyboardType: TextInputType.emailAddress,

        decoration: InputDecoration(

          hintText: 'Ingresa tu email...',

          labelText: 'Email',

          errorText: \_emailShowError ? \_emailError : null,

          prefixIcon: Icon(Icons.alternate\_email),

          suffixIcon: Icon(Icons.email),

          border: OutlineInputBorder(borderRadius: BorderRadius.circular(10)),

        ),

        onChanged: (value) {

          \_email = value;

        },

      ),

    );

  }

  Widget \_showButtons() {

    return Container(

      margin: EdgeInsets.only(left: 10, right: 10),

      child: Row(

        mainAxisAlignment: MainAxisAlignment.spaceAround,

        children: <Widget>[

          \_showRecoverButton(),

        ],

      ),

    );

  }

  Widget \_showRecoverButton() {

    return Expanded(

      child: ElevatedButton(

        child: Text('Recuperar Contraseña'),

        style: ButtonStyle(

          backgroundColor: MaterialStateProperty.resolveWith<Color>(

              (Set<MaterialState> states) {

            return Color(0xFF120E43);

          }),

        ),

        onPressed: () => \_recoverPassword(),

      ),

    );

  }

  void \_recoverPassword() {

    if (!\_validateFields()) {

      return;

    }

    \_sendRecoverPassword();

  }

  bool \_validateFields() {

    bool isValid = true;

    if (\_email.isEmpty) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar tu email.';

    } else if (!EmailValidator.validate(\_email)) {

      isValid = false;

      \_emailShowError = true;

      \_emailError = 'Debes ingresar un email válido.';

    } else {

      \_emailShowError = false;

    }

    setState(() {});

    return isValid;

  }

  void \_sendRecoverPassword() async {

    setState(() {

      \_showLoader = true;

    });

    var connectivityResult = await Connectivity().checkConnectivity();

    if (connectivityResult == ConnectivityResult.none) {

      setState(() {

        \_showLoader = false;

      });

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: 'Verifica que estes conectado a internet.',

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    Map<String, dynamic> request = {

      'email': \_email,

    };

    Response response = await ApiHelper.postNoToken(

      '/api/Account/RecoverPassword',

      request,

    );

    setState(() {

      \_showLoader = false;

    });

    if (!response.isSuccess) {

      await showAlertDialog(

          context: context,

          title: 'Error',

          message: response.message,

          actions: <AlertDialogAction>[

            AlertDialogAction(key: null, label: 'Aceptar'),

          ]);

      return;

    }

    await showAlertDialog(

        context: context,

        title: 'Confirmación',

        message:

            'Se le ha enviado un correo con las instrucciones para recuperar su contraseña',

        actions: <AlertDialogAction>[

          AlertDialogAction(key: null, label: 'Aceptar'),

        ]);

    Navigator.pop(context);

  }

}

En **login\_screen.dart** hacemos:

@override

  Widget build(BuildContext context) {

    return Scaffold(

      body: Stack(

        children: <Widget>[

          SingleChildScrollView(

            child: Column(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                SizedBox(

                  height: 40,

                ),

                \_showLogo(),

                SizedBox(

                  height: 20,

                ),

                \_showEmail(),

                \_showPassword(),

                \_showRememberme(),

                \_showForgotPassword(),

                \_showButtons(),

              ],

            ),

          ),

          \_showLoader

              ? LoaderComponent(text: 'Por favor espere...')

              : Container(),

        ],

      ),

    );

  }

 Widget \_showForgotPassword() {

    return InkWell(

      onTap: () => \_goForgotPassword(),

      child: Container(

        margin: EdgeInsets.only(bottom: 20),

        child: Text(

          '¿Has olvidado tu contraseña?',

          style: TextStyle(color: Colors.blue),

        ),

      ),

    );

  }

  void \_goForgotPassword() {

    Navigator.push(

      context,

      MaterialPageRoute(

        builder: (context) => RecoverPasswordScreen()

      )

    );

  }

# Mis Vehículos

En la **user\_info\_screen** hacemos:

class UserInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final bool isAdmin;

  UserInfoScreen(

      {required this.token, required this.user, required this.isAdmin});

y en la **users\_screen** hacemos:

void \_goInfoUser(User user) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) =>

                UserInfoScreen(token: widget.token, user: user, isAdmin:true)));

    if (result == 'yes') {

      \_getUsers();

    }

  }

En la **home\_screen** hacemos:

\_getCustomerMenu() {

    return Drawer(

      child: ListView(

        padding: EdgeInsets.zero,

        children: <Widget>[

          DrawerHeader(child: Image(image: AssetImage('assets/logo.png'))),

          ListTile(

            leading: Icon(Icons.two\_wheeler),

            title: Text('Mis Vehículos'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => UserInfoScreen(

                            token: widget.token,

                            user: widget.token.user,

                            isAdmin: false,

                          )));

                          },

          ),

          Divider(

            color: Colors.black,

            height: 2,

          ),

          ListTile(

            leading: Icon(Icons.face),

            title: const Text('Editar Perfil'),

            onTap: () {

              Navigator.push(

                  context,

                  MaterialPageRoute(

                      builder: (context) => User2Screen(

                            token: widget.token,

                            user: widget.token.user2,

                            myProfile: true,

                          )));

            },

          ),

          ListTile(

            leading: Icon(Icons.logout),

            title: Text('Cerrar Sesión'),

            onTap: () {

              \_logOut();

            },

          ),

        ],

      ),

    );

  }

## Deshabilitar Agregar Historias en la Vehicle\_Info\_Screen para un Usuario

Agregamos en **vehicle\_info\_screen**:

class VehicleInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final bool isAdmin;

  VehicleInfoScreen(

      {required this.token, required this.user, required this.vehicle,required this.isAdmin});

En **user\_info\_screen** hacemos:

void \_goVehicle(Vehicle vehicle) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => VehicleInfoScreen(

                token: widget.token, user: \_user, vehicle: vehicle,isAdmin:widget.isAdmin,)));

    if (result == 'yes') {

      \_getUser();

    }

  }

En **vehicle\_info\_screen** hacemos:

@override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

            '${\_vehicle.brand.description} ${\_vehicle.line} ${\_vehicle.plaque}'),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(

                text: 'Por favor espere...',

              )

            : \_getContent(),

      ),

      floatingActionButton: widget.isAdmin

      ? FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goAddHistory(History(

            date: '',

            dateLocal: '',

            details: [],

            detailsCount: 0,

            id: 0,

            mileage: 0,

            remarks: '',

            total: 0,

            totalLabor: 0,

            totalSpareParts: 0)),

      )

      : Container(),

    );

  }

También debemos deshabilitar el botón flotante en la **history\_info\_screen**:

class HistoryInfoScreen extends StatefulWidget {

  final Token token;

  final User user;

  final Vehicle vehicle;

  final History history;

  final bool isAdmin;

  HistoryInfoScreen(

      {required this.token,

      required this.user,

      required this.vehicle,

      required this.history,required this.isAdmin});

En **vehicle\_info\_screen** hacemos:

void \_goHistory(History history) async {

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => HistoryInfoScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: \_vehicle,

                  history: history,

                  isAdmin: widget.isAdmin,

                )));

    if (result == 'yes') {

      await \_getVehicle();

    }

  }

Y en **history\_info\_screen** hacemos:

@override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(

            '${widget.vehicle.brand.description} ${widget.vehicle.line} ${widget.vehicle.plaque}'),

      ),

      body: Center(

        child: \_showLoader

            ? LoaderComponent(

                text: 'Por favor espere...',

              )

            : \_getContent(),

      ),

      floatingActionButton: widget.isAdmin

          ? FloatingActionButton(

        child: Icon(Icons.add),

        onPressed: () => \_goDetail(Detail(

            id: 0,

            procedure: Procedure(id: 0, description: '', price: 0),

            laborPrice: 0,

            sparePartsPrice: 0,

            totalPrice: 0,

            remarks: '')),

      ): Container(),

    );

  }

 widget.isAdmin ?

              Positioned(

                  bottom: 0,

                  left: 280,

                  child: InkWell(

                    onTap: () => \_goEditHistory(),

                    child: ClipRRect(

                      borderRadius: BorderRadius.circular(30),

                      child: Container(

                        color: Colors.green[50],

                        height: 40,

                        width: 40,

                        child: Icon(

                          Icons.edit,

                          size: 30,

                          color: Colors.blue,

                        ),

                      ),

                    ),

                  )):Container()

            ],

void \_goDetail(Detail detail) async {

    if (!widget.isAdmin) {

      return;

    }

    String? result = await Navigator.push(

        context,

        MaterialPageRoute(

            builder: (context) => DetailScreen(

                  token: widget.token,

                  user: widget.user,

                  vehicle: widget.vehicle,

                  history: widget.history,

                  detail: detail,

                )));

    if (result == 'yes') {

      await \_getHistory();

    }

  }

widget.isAdmin ?

                  Icon(

                    Icons.arrow\_forward\_ios,

                    size: 40,

                  ) : Container()

# Política de Privacidad en el Back End

En \_**Layout.cshtml** hacemos

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2021 - Vehicles - <a **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Política de Privacidad</a>

</div>

</footer>

En Views/Home modificamos **Privacy.cshtml**

@{

ViewData["Title"] = "Política de Privacidad";

}

<h1>@ViewData["Title"]</h1>

<p>

Vehicles dando cumplimiento de las normas contenidas en la

Ley 1581 de 2012, en el Decreto 1377 de 2013 y en las demás normas concordantes por las

cuales se dictan disposiciones generales para la protección de datos personales, y en vista

de su calidad de Responsable del Tratamiento de Datos Personales de conformidad con la

aludida normatividad, se permite dar a conocer la presente Política de Privacidad y de

Protección de Datos Personales (en adelante la “Política”) para regular la recolección,

almacenamiento, tratamiento, administración, transferencia, transmisión, protección y

supresión de aquella información que se reciba de los titulares de datos personales o de

terceros a través de los diferentes canales de recolección de datos que ha dispuesto en el

desarrollo de sus actividades.

</p>

<strong>Definiciones</strong>

<p>

De acuerdo con los conceptos y criterios de la Ley Estatutaria 1581 de 2012 y el decreto

reglamentario 1377 de 2103 y/o para efectos de la presente Política, las palabras que a

continuación se definen tendrán el significado asignado en este capítulo, sea que se escriban

o no en mayúsculas, o que se encuentren en plural o singular, los mismos serán

desarrollados y aplicados bajo una interpretación sistemática e integral establecidos en la

mencionada normatividad

</p>

<p>

i) Aviso de privacidad: Comunicación verbal o escrita generada por el Responsable,

dirigida al Titular para el Tratamiento de sus datos personales, mediante la cual se le

informa acerca de la existencia de las políticas de Tratamiento de información que le serán

aplicables, la forma de acceder a las mismas y las finalidades del Tratamiento que se

pretende dar a los datos personales.

</p>

<p>

ii) Autorización: Consentimiento previo, expreso e informado del Titular para llevar a cabo

el Tratamiento de Datos Personales.

</p>

<p>

iii) Base de Datos: Conjunto organizado de Datos Personales que sea objeto de

Tratamiento, los cuales pueden ser almacenados y/o procesados en servidores ubicados en

centros de cómputo o papelería física, ya sean propios o contratados con terceros,

localizados en el territorio nacional o en distintos países.

</p>

<p>

iv) Dato Personal: Cualquier información vinculada o que pueda asociarse a una o varias

personas naturales determinadas o determinables.

</p>

<p>

v) Dato publico: es el dato que no sea semiprivado, privado o sensible. Son considerados

datos públicos, entre otros, los datos relativos al estado civil de las personas, a su profesión

u oficio y a su calidad de comerciante o de servidor publico. Por su naturaleza, los datos

públicos pueden estar contenidos, entre otros, en registros públicos, documentos públicos,

gacetas y boletines oficiales y sentencias judiciales debidamente ejecutoriadas que no estén

sometidas a reserva.

</p>

<p>

vi) Datos Sensibles: Se entiende por datos sensibles aquellos que afectan la intimidad del

Titular o cuyo uso indebido puede generar su discriminación, tales como aquellos que

revelen el origen racial o étnico, la orientación política, las convicciones religiosas o

filosóficas, la pertenencia a sindicatos, organizaciones sociales, de derechos humanos o que

promueva intereses de cualquier partido político o que garanticen los derechos y garantías

de partidos políticos de oposición así como los datos relativos a la salud, a la vida sexual y

los datos biométricos, entre otros.

</p>

<p>

vii) Encargado del Tratamiento: Persona natural o jurídica, pública o privada, que por sí

misma o en asocio con otros, realice el Tratamiento de Datos Personales por cuenta del

Responsable del Tratamiento de Datos Personales.

</p>

<strong>Principios</strong>

<p>

En el desarrollo, interpretación y aplicación de la ley 1581 de dos mil doce (2012) por la

cual se dictan disposiciones generales para la protección de datos personales y las normas

que la complementan, modifican o adicionan, se aplicarán de manera armónica e integral

los siguientes principios rectores:

</p>

<p>

Principio De Finalidad: el tratamiento debe obedecer a una finalidad legítima de acuerdo

con la Constitución y la Ley, la cual debe ser informada al titular.

</p>

<p>

Principio De Libertad: el tratamiento solo puede ejercerse con el consentimiento previo,

expreso, e informado del titular. Los datos personales no podrán ser obtenidos o divulgados

sin previa autorización, o en ausencia de mandato legal o judicial que releve el

consentimiento.

</p>

<p>

Principio De Veracidad O Calidad: la información sujeta a tratamiento debe ser veraz,

completa, exacta, actualizada, comprobable y comprensible. Se prohíbe el tratamiento de

datos parciales, incompletos, fraccionados o que induzcan a error.

</p>

<p>

Principio De Transparencia: en el tratamiento debe garantizarse el derecho del titular a

obtener del responsable del tratamiento o del encargado del tratamiento, en cualquier

momento y sin restricciones, información acerca de la existencia de datos que le

conciernan.

</p>

<p>

Principio De Acceso Y Circulación Restringida: el tratamiento se sujeta a los límites que se

derivan de la naturaleza de los datos personales, de las disposiciones de la ley y la

Constitución. En este sentido, el tratamiento sólo podrá hacerse por personas autorizadas

por el titular y/o por las personas previstas por la ley.

</p>

<p>

Los datos personales, salvo la información pública, no podrán estar disponibles en internet

u otros medios de divulgación o comunicación masiva, salvo que el acceso sea

técnicamente controlable para brindar un conocimiento restringido sólo a los titulares o

terceros autorizados.

</p>

<p>

Principio De Seguridad: la información sujeta a tratamiento por Gloria Colombia S.A, se

deberá manejar con las medidas técnicas, humanas y administrativas que sean necesarias

para otorgar seguridad a los registros evitando su adulteración, pérdida, consulta, uso o

acceso no autorizado o fraudulento.

</p>

# Menú Administradores

En \_**Layout.cshtml** agregamos:

@if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Brands" **asp-action**="Index">Marcas</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Users" **asp-action**="Index2">Mecánicos</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Procedures" **asp-action**="Index">Procedimientos</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="DocumentTypes" **asp-action**="Index">Tipos de Documento</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="VehicleTypes" **asp-action**="Index">Tipos de Vehículo</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Users" **asp-action**="Index">Usuarios</a>

</li>

}

En **UserController** agregamos:

//\*\*\*\*\*\*\*\*\*\* MECANICOS Index \*\*\*\*\*\*\*\*\*\*

public async Task<IActionResult> Index2()

{

return View(await \_context.Users

.Include(x => x.DocumentType)

.Include(x => x.Vehicles)

.Where(x => x.UserType == UserType.Admin)

.ToListAsync());

}

//\*\*\*\*\*\*\*\*\*\* MECANICOS Crear \*\*\*\*\*\*\*\*\*\*

public IActionResult Create2()

{

UserViewModel model = new UserViewModel

{

DocumentTypes = \_combosHelper.GetComboDocumentTypes()

};

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create2(UserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = string.Empty;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_converterHelper.ToUserAsync(model, imageId, true);

user.UserType = UserType.Admin;

await \_userHelper.AddUserAsync(user, "123456");

await \_userHelper.AddUserToRoleAsync(user, user.UserType.ToString());

string myToken = await \_userHelper.GenerateEmailConfirmationTokenAsync(user);

string tokenLink = Url.Action("ConfirmEmail", "Account", new

{

userid = user.Id,

token = myToken

}, protocol: HttpContext.Request.Scheme);

Response response = \_mailHelper.SendMail(model.Email, "Vehicles - Confirmación de cuenta", $"<h1>Vehicles - Confirmación de cuenta</h1>" +

$"Para habilitar el usuario, " +

$"por favor hacer clic en el siguiente enlace: </br></br><a href = \"{tokenLink}\">Confirmar Email</a>");

return RedirectToAction(nameof(Index2));

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

//\*\*\*\*\*\*\*\*\*\* MECANICOS Editar \*\*\*\*\*\*\*\*\*\*

public async Task<IActionResult> Edit2(string id)

{

if (string.IsNullOrEmpty(id))

{

return NotFound();

}

User user = await \_userHelper.GetUserByIdAsync(id);

if (user == null)

{

return NotFound();

}

UserViewModel model = \_converterHelper.ToUserViewModel(user);

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit2(UserViewModel model)

{

if (ModelState.IsValid)

{

string imageId = model.ImageId;

if (model.ImageFile != null)

{

imageId = await \_imageHelper.UploadImageAsync(model.ImageFile, "users");

}

User user = await \_converterHelper.ToUserAsync(model, imageId, false);

await \_userHelper.UpdateUserAsync(user);

return RedirectToAction(nameof(Index2));

}

model.DocumentTypes = \_combosHelper.GetComboDocumentTypes();

return View(model);

}

//\*\*\*\*\*\*\*\*\*\* MECANICOS Borrar \*\*\*\*\*\*\*\*\*\*

public async Task<IActionResult> Delete2(string id)

{

if (string.IsNullOrEmpty(id))

{

return NotFound();

}

User user = await \_userHelper.GetUserByIdAsync(id);

if (user == null)

{

return NotFound();

}

await \_userHelper.DeleteUserAsync(user);

return RedirectToAction(nameof(Index2));

}

Y la vista **Index2**:

@model IEnumerable<Vehicles.Api.Data.Entities.User>

@{

ViewData["Title"] = "Index2";

}

<link rel="stylesheet" href="https://cdn.datatables.net/1.10.19/css/jquery.dataTables.min.css" />

<p>

<a **asp-action**="Create2" class="btn btn-primary">Nuevo</a>

</p>

<div class="row">

<div class="col-md-12">

<div class="panel panel-default">

<div class="panel-heading">

<h3 class="panel-title">Mecánicos</h3>

</div>

<div class="panel-body">

<table class="table table-hover table-responsive table-striped" id="MyTable">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.FullName)

</th>

<th>

@Html.DisplayNameFor(model => model.DocumentType.Description)

</th>

<th>

@Html.DisplayNameFor(model => model.Document)

</th>

<th>

@Html.DisplayNameFor(model => model.Email)

</th>

<th>

@Html.DisplayNameFor(model => model.PhoneNumber)

</th>

<th>

@Html.DisplayNameFor(model => model.VehiclesCount)

</th>

<th>

@Html.DisplayNameFor(model => model.ImageFullPath)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.FullName)

</td>

<td>

@Html.DisplayFor(modelItem => item.DocumentType.Description)

</td>

<td>

@Html.DisplayFor(modelItem => item.Document)

</td>

<td>

@Html.DisplayFor(modelItem => item.Email)

</td>

<td>

@Html.DisplayFor(modelItem => item.PhoneNumber)

</td>

<td>

@Html.DisplayFor(modelItem => item.VehiclesCount)

</td>

<td>

<img src="@item.ImageFullPath" style="width: 100px; height: 100px; border-radius: 150px;" />

</td>

<td>

<a **asp-action**="Edit2" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

</tr>

}

</tbody>

</table>

</div>

</div>

</div>

</div>

<**partial** **name**="\_DeleteDialog" />

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

<script src="//cdn.datatables.net/1.10.19/js/jquery.dataTables.min.js"></script>

<script src="/js/deleteDialog.js"></script>

<script type="text/javascript">

$(document).ready(function () {

$('#MyTable').DataTable({

"language": {

"url": "//cdn.datatables.net/plug-ins/9dcbecd42ad/i18n/Spanish.json"

},

"aLengthMenu": [

[25, 50, 100, 200, -1],

[25, 50, 100, 200, "Todos"]

]

});

// Delete item

sc\_deleteDialog.openModal('deleteItem', true, 'btnYesDelete', '/Users/Delete2/', false);

});

</script>

}

Y la vista **Create2**:

@model Vehicles.Api.Models.UserViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Crear</h2>

<h4>Mecánico</h4>

<hr />

<div class="row">

<div class="col-md-12">

<form **asp-action**="Create2" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label **asp-for**="Email" class="control-label"></label>

<input **asp-for**="Email" class="form-control" />

<span **asp-validation-for**="Email" class="text-danger"></span>

</div>

<**partial** **name**="\_User" />

<div class="form-group">

<input type="submit" value="Crear" class="btn btn-primary" />

<a **asp-action**="Index2" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

Y la vista **Edit2**:

@model Vehicles.Api.Models.UserViewModel

@{

ViewData["Title"] = "Create";

}

<h2>Editar</h2>

<h4>Mecánico</h4>

<hr />

<div class="row">

<div class="col-md-10">

<form **asp-action**="Edit2" enctype="multipart/form-data">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<input **type**="hidden" **asp-for**="Id" />

<input **type**="hidden" **asp-for**="ImageId" />

<input **type**="hidden" **asp-for**="Email" />

<**partial** **name**="\_User" />

<div class="form-group">

<input type="submit" value="Guardar" class="btn btn-primary" />

<a **asp-action**="Index2" class="btn btn-success">Regresar</a>

</div>

</form>

</div>

<div class="col-md-2">

<img src="@Model.ImageFullPath" style="width: 200px; height: 200px; border-radius: 150px;" />

</div>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

# Llamar por teléfono al Taller

Vamos a la página

<https://pub.dev/>

Buscamos **url\_launcher**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit flutter pub get):

dependencies:

url\_launcher: ^6.0.12

Copiamos la línea

url\_launcher: ^6.0.12

y la pegamos en el archivo **pubspec.yaml**

Vamos a la página

<https://pub.dev/>

Buscamos **whatsapp\_unilink**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

dependencies:

whatsapp\_unilink: ^2.0.0

Copiamos la línea

whatsapp\_unilink: ^2.0.0

y la pegamos en el archivo **pubspec.yaml**

En **home\_screen** hacemos:

Widget \_getBody() {

    return SingleChildScrollView(

      child: Container(

        margin: EdgeInsets.all(30),

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            ClipRRect(

                borderRadius: BorderRadius.circular(150),

                child: CachedNetworkImage(

                  imageUrl: widget.token.user2.imageFullPath,

                  errorWidget: (context, url, error) => Icon(Icons.error),

                  fit: BoxFit.cover,

                  height: 300,

                  width: 300,

                  placeholder: (context, url) => Image(

                    image: AssetImage('assets/vehicles\_logo.png'),

                    fit: BoxFit.cover,

                    height: 300,

                    width: 300,

                  ),

                )),

            SizedBox(

              height: 30,

            ),

            Center(

              child: Text(

                'Bienvenid@ ${widget.token.user2.fullName}',

                style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

              ),

            ),

            SizedBox(

              height: 10,

            ),

            Row(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                Text('Llamar al taller'),

                SizedBox(

                  width: 10,

                ),

                ClipRRect(

                  borderRadius: BorderRadius.circular(20),

                  child: Container(

                    height: 40,

                    width: 40,

                    color: Colors.blue,

                    child: IconButton(

                      icon: Icon(

                        Icons.call,

                        color: Colors.white,

                      ),

                      onPressed: () => launch("tel://3516814963"),

                    ),

                  ),

                )

              ],

            ),

            SizedBox(

              height: 10,

            ),

            Row(

              mainAxisAlignment: MainAxisAlignment.center,

              children: <Widget>[

                Text('Enviar mensaje al taller'),

                SizedBox(

                  width: 10,

                ),

                ClipRRect(

                  borderRadius: BorderRadius.circular(20),

                  child: Container(

                    height: 40,

                    width: 40,

                    color: Colors.green,

                    child: IconButton(

                      icon: Icon(

                        Icons.insert\_comment,

                        color: Colors.white,

                      ),

                      onPressed: () => \_sendMessage(),

                    ),

                  ),

                )

              ],

            ),

          ],

        ),

      ),

    );

  }

void \_sendMessage() async {

    final link = WhatsAppUnilink(

      phoneNumber: '3516814963',

      text: 'Hola soy ${widget.token.user2.fullName} cliente del taller',

    );

    await launch('$link');

  }

# Llamar por teléfono a un usuario

En **user\_info\_screen** hacemos:

Row(

                          children: <Widget>[

                            Text('N° Vehículos: ',

                                style: TextStyle(fontWeight: FontWeight.bold)),

                            Text(

                              \_user.vehiclesCount.toString(),

                              style: TextStyle(

                                fontSize: 14,

                              ),

                            ),

                          ],

                        ),

                        SizedBox(height: 5,),

                        widget.isAdmin ? \_showCallButtons() : Container()

                      ],

                    ),

                  ),

                ],

              ),

            ),

          ),

        ],

      ),

    );

  }

Widget \_showCallButtons() {

    return Row(

      mainAxisAlignment: MainAxisAlignment.end,

      children: <Widget>[

          ClipRRect(

            borderRadius: BorderRadius.circular(20),

            child: Container(

              height: 40,

              width: 40,

              color: Colors.blue,

              child: IconButton(

                icon: Icon(Icons.call, color: Colors.white,),

                onPressed: () => launch('tel://${widget.user.phoneNumber}'),

              ),

            ),

          ),

          SizedBox(width: 10,),

          ClipRRect(

            borderRadius: BorderRadius.circular(20),

            child: Container(

              height: 40,

              width: 40,

              color: Colors.green,

              child: IconButton(

                icon: Icon(Icons.insert\_comment, color: Colors.white,),

                onPressed: () => \_sendMessage(),

              ),

            ),

          ),

          SizedBox(width: 10,),

      ],

    );

  }

  void \_sendMessage() async {

    final link = WhatsAppUnilink(

      phoneNumber: '${widget.user.phoneNumber}',

      text: 'Hola te escribo del taller.',

    );

    await launch('$link');

  }

# Menú Mis Vehículos en el BackEnd

En **\_Layout.cshtml** hacemos:

@if (User.Identity.IsAuthenticated && User.IsInRole("Admin"))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Brands" **asp-action**="Index">Marcas</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Users" **asp-action**="Index2">Mecánicos</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Procedures" **asp-action**="Index">Procedimientos</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="DocumentTypes" **asp-action**="Index">Tipos de Documento</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="VehicleTypes" **asp-action**="Index">Tipos de Vehículo</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Users" **asp-action**="Index">Usuarios</a>

</li>

}

@if (User.Identity.IsAuthenticated && User.IsInRole("User"))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Users" **asp-action**="MyVehicles">Mis Vehículos</a>

</li>

}

En **UsersControllers** hacemos

[Authorize(Roles = "Admin,User")]

public class UsersController : Controller

{

private readonly DataContext \_context;

private readonly IUserHelper \_userHelper;

private readonly ICombosHelper \_combosHelper;

private readonly IConverterHelper \_converterHelper;

private readonly IImageHelper \_imageHelper;

private readonly IMailHelper \_mailHelper;

public UsersController(DataContext context, IUserHelper userHelper, ICombosHelper combosHelper, IConverterHelper converterHelper, IImageHelper imageHelper,IMailHelper mailHelper)

{

\_context = context;

\_userHelper = userHelper;

\_combosHelper = combosHelper;

\_converterHelper = converterHelper;

\_imageHelper = imageHelper;

\_mailHelper = mailHelper;

}

//\*\*\*\*\*\*\*\*\*\* VEHICULOS del Usuario logueado \*\*\*\*\*\*\*\*\*\*

public async Task<IActionResult> MyVehicles()

{

User user = await \_userHelper.GetUserAsync(User.Identity.Name);

if (user == null)

{

return NotFound();

}

return RedirectToAction(nameof(Details), new { id = user.Id });

}

En la vista **DetailsVehicle** hacemos:

<div>

@if (User.IsInRole("Admin"))

{

<a **asp-action**="AddHistory" **asp-route-id**="@Model.Id" class="btn btn-primary">Adicionar Historia</a>

}

<a **asp-action**="EditVehicle" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar Vehículo</a>

<a **asp-action**="Details" **asp-route-id**="@Model.User.Id" class="btn btn-success">Regresar</a>

</div>

<td>

@if (User.IsInRole("Admin"))

{

<a **asp-action**="EditHistory" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

}

<a **asp-action**="DetailsHistory" **asp-route-id**="@item.Id" class="btn btn-info">Detalle<i class="glyphicon glyphicon-align-justify"></i></a>

@if (User.IsInRole("Admin"))

{

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

}

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

</td>

En la vista **DetailsHistory** hacemos:

<div>

@if (User.IsInRole("Admin"))

{

<a **asp-action**="AddDetail" **asp-route-id**="@Model.Id" class="btn btn-primary">Adicionar Procedimiento</a>

<a **asp-action**="EditHistory" **asp-route-id**="@Model.Id" class="btn btn-warning">Editar Historia</a>

}

<a **asp-action**="DetailsVehicle" **asp-route-id**="@Model.Vehicle.Id" class="btn btn-success">Regresar</a>

</div>

<td>

@if (User.IsInRole("Admin"))

{

<a **asp-action**="EditDetail" **asp-route-id**="@item.Id" class="btn btn-warning">Editar</a>

<button data-id="@item.Id" class="btn btn-danger deleteItem" data-toggle="modal" data-target="#deleteDialog">Borrar</button>

}

</td>

# Publicación en Google Play

## Agregar un ícono

Generamos el ícono en Android Asset Studio.

Obtenemos además una imagen de 1024 x 1024 y debemos generar otra de 1024 x 500 pixeles.

Vamos a la página

<https://pub.dev/>

Buscamos **flutter\_launcher\_icons**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

dependencies:

flutter\_launcher\_icons: ^0.9.2

Copiamos la línea

flutter\_launcher\_icons: ^0.9.2

y la pegamos en el archivo **pubspec.yaml**

Agregamos a continuación en el archivo **pubspec.yaml**

flutter\_icons:

  android: "launcher\_icon"

  ios: true

  image\_path: "assets/icon/icon.png"

Copiamos desde la carpeta **res\mipmap-xxxhdpi** el archivo **ic\_launcher.png** y lo pegamos en la carpeta **assets**

Corregimos en **pubspec.yaml** el nombre del archivo:

flutter\_icons:

  android: "launcher\_icon"

  ios: true

  image\_path: "assets/icon/ic\_launcher.png"

Corremos en la Terminal el comando

flutter pub run flutter\_launcher\_icons:main

## Splash

Vamos a la página

<https://pub.dev/>

Buscamos **flutter\_native\_splash**

Lo seleccionamos

Vamos a la pestaña **Installing** y ahí dice:

This will add a line like this to your package's pubspec.yaml (and run an implicit dart pub get):

dependencies:

flutter\_native\_splash: ^1.2.4

Copiamos la línea

flutter\_native\_splash: ^1.2.4

y la pegamos en el archivo **pubspec.yaml**

Agregamos a continuación en el archivo **pubspec.yaml**

flutter\_native\_splash:

  image: "assets/logo.png"

  color: "#FFFFFF"

Corremos en la Terminal el comando

flutter pub run flutter\_native\_splash:create

Me posiciono en **VEHICLES** y busco **com.example**

** **

Y cambio por **com.luisnu.vehicles**

Asegurarse que el Java JRE esté en el Path en las variables de ambiente.

## Firmar la Aplicación

Dentro de la carpeta **lib** creamos un archivo que llamamos **key.txt**

Ahí ponemos:

keytool -genkey -v -keystore c:\Users\USER\_NAME\upload-keystore.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias upload

En el directorio raíz del Proyecto creamos una carpeta **keys**

Modificamos en **key.txt**

keytool -genkey -v -keystore keys\key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias sd

alias: sd

pass: 123456

Copiamos el comando

keytool -genkey -v -keystore keys\key.jks -storetype JKS -keyalg RSA -keysize 2048 -validity 10000 -alias sd

y lo pegamos en una ventana cmd en la carpeta del proyecto **F:\Flutter\vehicles\_app**

Hace una serie de preguntas. Las contestamos. Al final nos recomienda

Warning:

El almacén de claves JKS utiliza un formato privativo. Se recomienda migrar a PKCS12, que es un formato estándar del sector que utiliza "keytool -importkeystore -srckeystore keys\key.jks -destkeystore keys\key.jks -deststoretype pkcs12".

Entonces ponemos en la ventana cmd

keytool -importkeystore -srckeystore keys\key.jks -destkeystore keys\key.jks -deststoretype pkcs12

Así se ha generado en la carpeta **keys**:



## Generar archivo aab

En **VEHICLES/Android/app/build.gradle** buscamos donde dice **android** y antes pegamos

def keystoreProperties = new Properties()

   def keystorePropertiesFile = rootProject.file('key.properties')

   if (keystorePropertiesFile.exists()) {

       keystoreProperties.load(new FileInputStream(keystorePropertiesFile))

   }

android {

    compileSdkVersion 30

    sourceSets {

        main.java.srcDirs += 'src/main/kotlin'

    }

También buscamos la sección que dice **buildTypes**

 buildTypes {

        release {

            // TODO: Add your own signing config for the release build.

            // Signing with the debug keys for now, so `flutter run --release` works.

            signingConfig signingConfigs.debug

        }

    }

Y la reemplazamos por estas líneas:

signingConfigs {

       release {

           keyAlias keystoreProperties['keyAlias']

           keyPassword keystoreProperties['keyPassword']

           storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) : null

           storePassword keystoreProperties['storePassword']

       }

   }

   buildTypes {

       release {

           signingConfig signingConfigs.release

       }

   }

Antes de crear el archivo aaab fijarse la versión en el archivo **pubspec.yaml**

version: 1.0.0+1

Hay que ir incrementando el número de versión cada vez.

En la Terminal ponemos el comando

flutter build appbundle

y así tenemos el archivo aab en la carpeta

/build/app/outputs/bundle/release/app.aab.