Namespace AlquilerMicroservicio.API.Controllers

Classes

RentalsController

Controlador HTTP que gestiona el alquiler y la devoluc d vehiculos, manda los comandos al mediador pa q los procese.

VehiclesController

Controlador que gestiona los vehículos. Aquí se crean y se listan los que están libres.

Class RentalsController

Namespace: AlquilerMicroservicio.API.Controllers

Assembly: AlquilerMicroservicio.API.dll

Controlador HTTP que gestiona el alquiler y la devoluc d vehiculos, manda los comandos al mediador pa q los procese.

```
[ApiController]
[Route("api/[controller]")]
public class RentalsController : ControllerBase
```

Inheritance

object ← ControllerBase ← RentalsController

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) , ControllerBase.Content(string) , ControllerBase.Content(string, string) , ControllerBase.Content(string, string, Encoding) , ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() , ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) , ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) , ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) , ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) , ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() , ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) , ControllerBase.RedirectToAction(string, string) , ControllerBase.RedirectToAction(string, string, object) , ControllerBase.RedirectToAction(string, string, string) , ControllerBase.RedirectToAction(string, string, object, string) , ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) , ControllerBase.RedirectToActionPermanent(string) , ControllerBase.RedirectToActionPermanent(string, object) , ControllerBase.RedirectToActionPermanent(string, string) , ControllerBase.RedirectToActionPermanent(string, string, string) , ControllerBase.RedirectToActionPermanent(string, string, object) , ControllerBase.RedirectToActionPermanent(string, string, object, string) , ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) , ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) ,

ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,
ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,

ControllerBase.PhysicalFile(string, string, string, bool) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary, IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,
ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() ,
ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,

ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() , ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RentalsController(IMediator)

Constructor que inyecta el mediador pa poder mandar comandos desde los endpoints.

```
public RentalsController(IMediator mediator)
```

Parameters

`mediator` IMediator

Methods

RentVehicle(RentVehicleCommand)

Recibe una peticion para alquilar un vehiculo y lo manda como comando via MediatR.

```
[HttpPost]
public Task<IActionResult> RentVehicle(RentVehicleCommand command)
```

Parameters

`command` RentVehicleCommand

Returns

Task<IActionResult>

ReturnVehicle(ReturnVehicleCommand)

Se encarga de gestionar la devolucion del vehiculo, mandando el comando correspondiente.

```
[HttpPost("return")]
public Task<IActionResult> ReturnVehicle(ReturnVehicleCommand command)
```

Parameters

`command` ReturnVehicleCommand

Returns

Task<IActionResult>

Class VehiclesController

Namespace: AlquilerMicroservicio.API.Controllers

Assembly: AlquilerMicroservicio.API.dll

Controlador que gestiona los vehículos. Aquí se crean y se listan los que están libres.

```
[ApiController]
[Route("api/[controller]")]
public class VehiclesController : ControllerBase
```

Inheritance

object ← ControllerBase ← VehiclesController

Inherited Members

ControllerBase.StatusCode(int) , ControllerBase.StatusCode(int, object) , ControllerBase.Content(string) , ControllerBase.Content(string, string) , ControllerBase.Content(string, string, Encoding) , ControllerBase.Content(string, MediaTypeHeaderValue) , ControllerBase.NoContent() , ControllerBase.Ok() , ControllerBase.Ok(object) , ControllerBase.Redirect(string) , ControllerBase.RedirectPermanent(string) , ControllerBase.RedirectPreserveMethod(string) , ControllerBase.RedirectPermanentPreserveMethod(string) , ControllerBase.LocalRedirect(string) , ControllerBase.LocalRedirectPermanent(string) , ControllerBase.LocalRedirectPreserveMethod(string) , ControllerBase.LocalRedirectPermanentPreserveMethod(string) , ControllerBase.RedirectToAction() , ControllerBase.RedirectToAction(string) , ControllerBase.RedirectToAction(string, object) , ControllerBase.RedirectToAction(string, string) , ControllerBase.RedirectToAction(string, string, object) , ControllerBase.RedirectToAction(string, string, string) , ControllerBase.RedirectToAction(string, string, object, string) , ControllerBase.RedirectToActionPreserveMethod(string, string, object, string) , ControllerBase.RedirectToActionPermanent(string) , ControllerBase.RedirectToActionPermanent(string, object) , ControllerBase.RedirectToActionPermanent(string, string) , ControllerBase.RedirectToActionPermanent(string, string, string) , ControllerBase.RedirectToActionPermanent(string, string, object) , ControllerBase.RedirectToActionPermanent(string, string, object, string) , ControllerBase.RedirectToActionPermanentPreserveMethod(string, string, object, string) , ControllerBase.RedirectToRoute(string) , ControllerBase.RedirectToRoute(object) , ControllerBase.RedirectToRoute(string, object) , ControllerBase.RedirectToRoute(string, string) ,

ControllerBase.RedirectToRoute(string, object, string) ,
ControllerBase.RedirectToRoutePreserveMethod(string, object, string) ,
ControllerBase.RedirectToRoutePermanent(string) ,
ControllerBase.RedirectToRoutePermanent(object) ,
ControllerBase.RedirectToRoutePermanent(string, object) ,
ControllerBase.RedirectToRoutePermanent(string, string) ,
ControllerBase.RedirectToRoutePermanent(string, object, string) ,
ControllerBase.RedirectToRoutePermanentPreserveMethod(string, object, string) ,
ControllerBase.RedirectToPage(string) , ControllerBase.RedirectToPage(string, object) ,
ControllerBase.RedirectToPage(string, string) , ControllerBase.RedirectToPage(string, string, object) ,
ControllerBase.RedirectToPage(string, string, string) ,
ControllerBase.RedirectToPage(string, string, object, string) ,
ControllerBase.RedirectToPagePermanent(string) ,
ControllerBase.RedirectToPagePermanent(string, object) ,
ControllerBase.RedirectToPagePermanent(string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, string) ,
ControllerBase.RedirectToPagePermanent(string, string, object, string) ,
ControllerBase.RedirectToPagePreserveMethod(string, string, object, string) ,
ControllerBase.RedirectToPagePermanentPreserveMethod(string, string, object, string) ,
ControllerBase.File(byte[], string) , ControllerBase.File(byte[], string, bool) ,
ControllerBase.File(byte[], string, string) , ControllerBase.File(byte[], string, string, bool) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(byte[], string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string) , ControllerBase.File(Stream, string, bool) ,
ControllerBase.File(Stream, string, string) , ControllerBase.File(Stream, string, string, bool) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(Stream, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string) , ControllerBase.File(string, string, bool) ,
ControllerBase.File(string, string, string) , ControllerBase.File(string, string, string, bool) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.File(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string) , ControllerBase.PhysicalFile(string, string, bool) ,
ControllerBase.PhysicalFile(string, string, string) ,
ControllerBase.PhysicalFile(string, string, string, bool) ,

ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue) ,
ControllerBase.PhysicalFile(string, string, string, DateTimeOffset?, EntityTagHeaderValue, bool) ,
ControllerBase.Unauthorized() , ControllerBase.Unauthorized(object) , ControllerBase.NotFound() ,
ControllerBase.NotFound(object) , ControllerBase.BadRequest() ,
ControllerBase.BadRequest(object) , ControllerBase.BadRequest(ModelStateDictionary) ,
ControllerBase.UnprocessableEntity() , ControllerBase.UnprocessableEntity(object) ,
ControllerBase.UnprocessableEntity(ModelStateDictionary) , ControllerBase.Conflict() ,
ControllerBase.Conflict(object) , ControllerBase.Conflict(ModelStateDictionary) ,
ControllerBase.Problem(string, string, int?, string, string) ,
ControllerBase.Problem(string, string, int?, string, string, IDictionary<string, object>) ,
ControllerBase.ValidationProblem(ValidationProblemDetails) ,
ControllerBase.ValidationProblem(ModelStateDictionary) , ControllerBase.ValidationProblem() ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary) ,
ControllerBase.ValidationProblem(string, string, int?, string, string, ModelStateDictionary, IDictionary<string, object>) ,
ControllerBase.Created() , ControllerBase.Created(string, object) ,
ControllerBase.Created(Uri, object) , ControllerBase.CreatedAtAction(string, object) ,
ControllerBase.CreatedAtAction(string, object, object) ,
ControllerBase.CreatedAtAction(string, string, object, object) ,
ControllerBase.CreatedAtRoute(string, object) , ControllerBase.CreatedAtRoute(object, object) ,
ControllerBase.CreatedAtRoute(string, object, object) , ControllerBase.Accepted() ,
ControllerBase.Accepted(object) , ControllerBase.Accepted(Uri) , ControllerBase.Accepted(string) ,
ControllerBase.Accepted(string, object) , ControllerBase.Accepted(Uri, object) ,
ControllerBase.AcceptedAtAction(string) , ControllerBase.AcceptedAtAction(string, string) ,
ControllerBase.AcceptedAtAction(string, object) ,
ControllerBase.AcceptedAtAction(string, string, object) ,
ControllerBase.AcceptedAtAction(string, object, object) ,
ControllerBase.AcceptedAtAction(string, string, object, object) ,
ControllerBase.AcceptedAtRoute(object) , ControllerBase.AcceptedAtRoute(string) ,
ControllerBase.AcceptedAtRoute(string, object) , ControllerBase.AcceptedAtRoute(object, object) ,
ControllerBase.AcceptedAtRoute(string, object, object) , ControllerBase.Challenge() ,
ControllerBase.Challenge(params string[]) , ControllerBase.Challenge(AuthenticationProperties) ,
ControllerBase.Challenge(AuthenticationProperties, params string[]) , ControllerBase.Forbid() ,
ControllerBase.Forbid(params string[]) , ControllerBase.Forbid(AuthenticationProperties) ,
ControllerBase.Forbid(AuthenticationProperties, params string[]) ,
ControllerBase.SignIn(ClaimsPrincipal) , ControllerBase.SignIn(ClaimsPrincipal, string) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties) ,
ControllerBase.SignIn(ClaimsPrincipal, AuthenticationProperties, string) , ControllerBase.SignOut() ,

ControllerBase.SignOut(AuthenticationProperties) , ControllerBase.SignOut(params string[]) , ControllerBase.SignOut(AuthenticationProperties, params string[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, params Expression<Func<TModel, object>>[]) , ControllerBase.TryUpdateModelAsync<TModel>(TModel, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryUpdateModelAsync(object, Type, string) , ControllerBase.TryUpdateModelAsync(object, Type, string, IValueProvider, Func<ModelMetadata, bool>) , ControllerBase.TryValidateModel(object) , ControllerBase.TryValidateModel(object, string) , ControllerBase.HttpContext , ControllerBase.Request , ControllerBase.Response , ControllerBase.RouteData , ControllerBase.ModelState , ControllerBase.ControllerContext , ControllerBase.MetadataProvider , ControllerBase.ModelBinderFactory , ControllerBase.Url , ControllerBase.ObjectValidator , ControllerBase.ProblemDetailsFactory , ControllerBase.User , ControllerBase.Empty , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

VehiclesController(IMediator)

Inyectamos el mediador para que esta cosa pueda enviar comandos y consultas.

```
public VehiclesController(IMediator mediator)
```

Parameters

`mediator` IMediator

Methods

CreateVehicle(CreateVehicleCommand)

Crea un vehículo nuevo. Lo manda al mediador y que él se apañe con lo que tenga que hacer.

```
[HttpPost]
public Task<IActionResult> CreateVehicle(CreateVehicleCommand command)
```

Parameters

`command` CreateVehicleCommand

Returns

Task<IActionResult>

GetAvailableVehicles()

Devuelve la lista de vehículos que no están pillados. Solo los disponibles.

```
[HttpGet("available")]
public Task<IActionResult> GetAvailableVehicles()
```

Returns

Task<IActionResult>

Namespace AlquilerMicroservicio.Application

Classes

ApplicationAssemblyReference

Referencia al ensamblado de la capa de aplicación. Se usa para registrar cosas como los handlers sin volverse loco buscando tipos.

ApplicationServiceCollectionExtensions

Clase de extensión que mete todo lo necesario de la capa de aplicación en el contenedor de dependencias.

Class ApplicationAssemblyReference

Namespace: AlquilerMicroservicio.Application

Assembly: AlquilerMicroservicio.Application.dll

Referencia al ensamblado de la capa de aplicación. Se usa para registrar cosas como los handlers sin volverse loco buscando tipos.

```
public static class ApplicationAssemblyReference
```

Inheritance

object ← ApplicationAssemblyReference

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Fields

Assembly

Ensamblado actual de la aplicación. Básicamente, apunta aquí para saber dónde está todo lo de esta capa.

```
public static readonly Assembly Assembly
```

Field Value

Assembly

Class ApplicationServiceCollectionExtensions

Namespace: AlquilerMicroservicio.Application

Assembly: AlquilerMicroservicio.Application.dll

Clase de extensión que mete todo lo necesario de la capa de aplicación en el contenedor de dependencias.

```
public static class ApplicationServiceCollectionExtensions
```

Inheritance

object ← ApplicationServiceCollectionExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

AddApplication(IServiceCollection)

Registra MediatR con todos los handlers que hay en esta capa. Lo deja listo para lanzar comandos y consultas.

```
public static IServiceCollection AddApplication(this IServiceCollection services)
```

Parameters

`services` IServiceCollection

Returns

IServiceCollection

Namespace AlquilerMicroservicio.Application.Commands

Classes

CreateVehicleCommand

Comando para crear un vehículo. Se lanza y que el handler se encargue del marrón.

RentVehicleCommand

Comando que se lanza cuando alguien quiere pillar un vehículo. Básicamente, reserva el trasto.

ReturnVehicleCommand

Comando que se lanza cuando el cliente trae el coche de vuelta.

Class CreateVehicleCommand

Namespace: AlquilerMicroservicio.Application.Commands

Assembly: AlquilerMicroservicio.Application.dll

Comando para crear un vehículo. Se lanza y que el handler se encargue del marrón.

```
public class CreateVehicleCommand : IRequest<Unit>, IBaseRequest
```

Inheritance

object ← CreateVehicleCommand

Implements

IRequest<Unit>, IBaseRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Brand

Marca del coche, moto o lo que sea que estemos registrando.

```
public string Brand { get; set; }
```

Property Value

string

Id

Identificador único del cacharro, que no se repita o la liamos.

```
public Guid Id { get; set; }
```

Property Value

Guid

ManufactureDate

Fecha en la que salió de fábrica. Si es muy viejo, igual ni arranca.

```
public DateTime ManufactureDate { get; set; }
```

Property Value

DateTime

Model

Modelo del vehículo, pa saber si es un trasto o algo decente.

```
public string Model { get; set; }
```

Property Value

string

Class RentVehicleCommand

Namespace: AlquilerMicroservicio.Application.Commands

Assembly: AlquilerMicroservicio.Application.dll

Comando que se lanza cuando alguien quiere pillar un vehículo. Básicamente, reserva el trasto.

```
public class RentVehicleCommand : IRequest<Unit>, IBaseRequest
```

Inheritance

object ← RentVehicleCommand

Implements

IRequest<Unit>, IBaseRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CustomerId

Id del cliente que hace la reserva. El que va a dar guerra.

```
public string CustomerId { get; set; }
```

Property Value

string

RentalId

Id del alquiler, pa tenerlo controlado y que no se mezcle con otro.

```
public Guid RentalId { get; set; }
```

Property Value

Guid

VehicleId

Id del vehículo que quieren llevarse.

```
public Guid VehicleId { get; set; }
```

Property Value

Guid

Class ReturnVehicleCommand

Namespace: AlquilerMicroservicio.Application.Commands

Assembly: AlquilerMicroservicio.Application.dll

Comando que se lanza cuando el cliente trae el coche de vuelta.

```
public class ReturnVehicleCommand : IRequest<Unit>, IBaseRequest
```

Inheritance

object ← ReturnVehicleCommand

Implements

IRequest<Unit>, IBaseRequest

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

RentalId

Id del alquiler que se quiere cerrar.

```
public Guid RentalId { get; set; }
```

Property Value

Guid

Namespace AlquilerMicroservicio.Application.EventHandlers

Classes

VehicleRentedDomainEventHandler

Manejador que se ejecuta cuando se alquila un vehículo. Básicamente, deja constancia del lío en los logs.

Class VehicleRentedDomainEventHandler

Namespace: AlquilerMicroservicio.Application.EventHandlers

Assembly: AlquilerMicroservicio.Application.dll

Manejador que se ejecuta cuando se alquila un vehículo. Básicamente, deja constancia del lío en los logs.

```
public class VehicleRentedDomainEventHandler :
INotificationHandler<VehicleRentedDomainEvent>
```

Inheritance

object ← VehicleRentedDomainEventHandler

Implements

INotificationHandler<VehicleRentedDomainEvent>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

VehicleRentedDomainEventHandler(ILogger<VehicleRentedDomainEventHandler>)

Se mete el logger para soltar la info cuando alguien alquila un vehículo.

```
public VehicleRentedDomainEventHandler(ILogger<VehicleRentedDomainEventHandler> logger)
```

Parameters

`logger` ILogger<VehicleRentedDomainEventHandler>

Methods

Handle(VehicleRentedDomainEvent, CancellationToken)

Maneja el evento de alquiler de vehículo y lo suelta en los logs pa que quede claro quién pilló qué.

```
public Task Handle(VehicleRentedDomainEvent notification, CancellationToken
cancellationToken)
```

Parameters

`notification` VehicleRentedDomainEvent

`cancellationToken` CancellationToken

Returns

Task

Namespace AlquilerMicroservicio.Application.Handlers

Classes

[CreateVehicleCommandHandler]

Handler que se encarga de registrar un vehículo nuevo. Lo mete en el repo y a otra cosa.

[GetAvailableVehiclesQueryHandler]

Handler que se encarga de buscar los vehículos que están libres.

[RentVehicleCommandHandler]

Handler que se encarga de alquilar un vehículo: usa el dominio, guarda lo que haya que guardar y lanza eventos.

[ReturnVehicleCommandHandler]

Handler que gestiona la devolución del vehículo. Revisa el alquiler y marca todo como devuelto.

Class CreateVehicleCommandHandler

Namespace: AlquilerMicroservicio.Application.Handlers

Assembly: AlquilerMicroservicio.Application.dll

Handler que se encarga de registrar un vehículo nuevo. Lo mete en el repo y a otra cosa.

```
public class CreateVehicleCommandHandler : IRequestHandler<CreateVehicleCommand, Unit>
```

Inheritance

object ← CreateVehicleCommandHandler

Implements

IRequestHandler<CreateVehicleCommand, Unit>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

CreateVehicleCommandHandler(IVehicleRepository)

Se inyecta el repositorio de vehículos.

```
public CreateVehicleCommandHandler(IVehicleRepository vehicleRepository)
```

Parameters

`vehicleRepository` IVehicleRepository

Methods

Handle(CreateVehicleCommand, CancellationToken)

Procesa el comando para crear un vehículo. Lo construye y lo suelta en el repositorio.

```
public Task<Unit> Handle(CreateVehicleCommand request, CancellationToken cancellationToken)
```

Parameters

`request` CreateVehicleCommand

`cancellationToken` CancellationToken

Returns

Task<Unit>

Class GetAvailableVehiclesQueryHandler

Namespace: AlquilerMicroservicio.Application.Handlers

Assembly: AlquilerMicroservicio.Application.dll

Handler que se encarga de buscar los vehículos que están libres.

```
public class GetAvailableVehiclesQueryHandler : IRequestHandler<GetAvailableVehiclesQuery,
List<Vehicle>>
```

Inheritance

object ← GetAvailableVehiclesQueryHandler

Implements

IRequestHandler<GetAvailableVehiclesQuery, List<Vehicle>>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

GetAvailableVehiclesQueryHandler(IVehicleRepository)

Se inyecta el repo pa poder ir a mirar qué vehículos siguen en el parking.

```
public GetAvailableVehiclesQueryHandler(IVehicleRepository vehicleRepository)
```

Parameters

`vehicleRepository` IVehicleRepository

Methods

Handle(GetAvailableVehiclesQuery, CancellationToken)

Maneja la consulta para traer los vehículos disponibles. Si no hay, pues se devuelve la lista vacía y ya.

```
public Task<List<Vehicle>> Handle(GetAvailableVehiclesQuery request,
CancellationToken cancellationToken)
```

Parameters

`request` GetAvailableVehiclesQuery

`cancellationToken` CancellationToken

Returns

Task<List<Vehicle>>

Class RentVehicleCommandHandler

Namespace: AlquilerMicroservicio.Application.Handlers

Assembly: AlquilerMicroservicio.Application.dll

Handler que se encarga de alquilar un vehículo: usa el dominio, guarda lo que haya que guardar y lanza eventos.

```
public class RentVehicleCommandHandler : IRequestHandler<RentVehicleCommand, Unit>
```

Inheritance

object ← RentVehicleCommandHandler

Implements

IRequestHandler<RentVehicleCommand, Unit>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RentVehicleCommandHandler(RentalDomainService, IRentalRepository, IVehicleRepository, IDomainEventDispatcher)

Aquí se mete todo lo necesario para poder alquilar: dominio, repos, eventos....

```
public RentVehicleCommandHandler(RentalDomainService rentalDomainService,
IRentalRepository rentalRepository, IVehicleRepository vehicleRepository,
IDomainEventDispatcher eventDispatcher)
```

Parameters

`rentalDomainService` RentalDomainService

`rentalRepository` IRentalRepository

`vehicleRepository` IVehicleRepository

`eventDispatcher` IDomainEventDispatcher

Methods

Handle(RentVehicleCommand, CancellationToken)

Procesa el alquiler del vehículo: se hace la lógica, se actualiza todo y se lanza el evento.

```
public Task<Unit> Handle(RentVehicleCommand request, CancellationToken cancellationToken)
```

Parameters

`request` RentVehicleCommand

`cancellationToken` CancellationToken

Returns

Task<Unit>

Class ReturnVehicleCommandHandler

Namespace: AlquilerMicroservicio.Application.Handlers

Assembly: AlquilerMicroservicio.Application.dll

Handler que gestiona la devolución del vehículo. Revisa el alquiler y marca todo como devuelto.

```
public class ReturnVehicleCommandHandler : IRequestHandler<ReturnVehicleCommand, Unit>
```

Inheritance

object ← ReturnVehicleCommandHandler

Implements

IRequestHandler<ReturnVehicleCommand, Unit>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ReturnVehicleCommandHandler(IRentalRepository, IVehicleRepository)

Inyectamos los repos necesarios pa buscar el alquiler y el coche, y actualizar el estado.

```
public ReturnVehicleCommandHandler(IRentalRepository rentalRepository,
IVehicleRepository vehicleRepository)
```

Parameters

`rentalRepository` IRentalRepository

`vehicleRepository` IVehicleRepository

Methods

Handle(ReturnVehicleCommand, CancellationToken)

Procesa la devolución: valida que el alquiler existe, que no está ya devuelto, y marca todo.

```
public Task<Unit> Handle(ReturnVehicleCommand request, CancellationToken cancellationToken)
```

Parameters

`request` ReturnVehicleCommand

`cancellationToken` CancellationToken

Returns

Task<Unit>

Namespace AlquilerMicroservicio.Application.Queries

Classes

GetAvailableVehiclesQuery

Query que pide la lista de vehículos disponibles.

Class GetAvailableVehiclesQuery

Namespace: AlquilerMicroservicio.Application.Queries

Assembly: AlquilerMicroservicio.Application.dll

Query que pide la lista de vehículos disponibles.

```
public class GetAvailableVehiclesQuery : IRequest<List<Vehicle>>, IBaseRequest
```

Inheritance

object ← GetAvailableVehiclesQuery

Implements

IRequest<List<Vehicle>>, IBaseRequest

Inherited Members

object.Equals(object), object.Equals(object, object), object.GetHashCode(), object.GetType(), object.MemberwiseClone(), object.ReferenceEquals(object, object), object.ToString()

Namespace AlquilerMicroservicio.Domain.Entities

Classes

[Rental](#)

Entidad que representa un alquiler. Guarda quién pilló qué coche, cuándo lo hizo y si ya lo devolvió o sigue alquilado.

[Vehicle](#)

Entidad que representa un vehículo con su información básica y estado de alquiler.

Class Rental

Namespace: AlquilerMicroservicio.Domain.Entities

Assembly: AlquilerMicroservicio.Domain.dll

Entidad que representa un alquiler. Guarda quién pilló qué coche, cuándo lo hizo y si ya lo devolvió o sigue alquilado.

```
public class Rental
```

Inheritance

object ← Rental

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Rental(Guid, Guid, string)

Crea una nueva instancia de alquiler con la info mínima. La fecha se mete sola al momento.

```
public Rental(Guid id, Guid vehicleId, string customerId)
```

Parameters

`id` Guid

`vehicleId` Guid

`customerId` string

Properties

CustomerId

Id del cliente que hizo el alquiler.

```
public string CustomerId { get; }
```

Property Value

string

Id

Identificador único del alquiler.

```
public Guid Id { get; }
```

Property Value

Guid

IsActive

Indica si el alquiler sigue activo.

```
public bool IsActive { get; }
```

Property Value

bool

RentalDate

Fecha y hora en la que se registró el alquiler. UTC para no liar husos.

```
public DateTime RentalDate { get; }
```

Property Value

DateTime

ReturnDate

Fecha de devolución, si ya lo trajo de vuelta. Si está a null, el coche sigue fuera.

```
public DateTime? ReturnDate { get; }
```

Property Value

DateTime?

VehicleId

Id del vehículo que fue alquilado. Aquí queda registrado.

```
public Guid VehicleId { get; }
```

Property Value

Guid

Methods

Return()

Marca el alquiler como devuelto. Si ya estaba cerrado, lanza excepción.

```
public void Return()
```

Class Vehicle

Namespace: AlquilerMicroservicio.Domain.Entities

Assembly: AlquilerMicroservicio.Domain.dll

Entidad que representa un vehículo con su información básica y estado de alquiler.

```
public class Vehicle
```

Inheritance

object ← Vehicle

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

Vehicle(Guid, string, string, DateTime)

Constructor que inicializa el vehículo. No permite registrar vehículos con más de 5 años.

```
public Vehicle(Guid id, string brand, string model, DateTime manufactureDate)
```

Parameters

`id` Guid

`brand` string

`model` string

`manufactureDate` DateTime

Properties

Brand

Marca del vehículo.

```
public string Brand { get; }
```

Property Value

string

Id

Identificador único del vehículo.

```
public Guid Id { get; }
```

Property Value

Guid

IsRented

Indica si el vehículo está alquilado o no.

```
public bool IsRented { get; }
```

Property Value

bool

ManufactureDate

Fecha de fabricación del vehículo.

```
public DateTime ManufactureDate { get; }
```

Property Value

[DateTime](#)

Model

Modelo del vehículo.

```
public string Model { get; }
```

Property Value

[string](#)

Methods

Rent()

Marca el vehículo como alquilado. Lanza excepción si ya lo está.

```
public void Rent()
```

Return()

Marca el vehículo como disponible. Lanza excepción si no estaba alquilado.

```
public void Return()
```

Namespace AlquilerMicroservicio.Domain.Events

Classes

VehicleRentedDomainEvent

Evento de dominio que se lanza cuando un vehículo ha sido alquilado.

Class VehicleRentedDomainEvent

Namespace: AlquilerMicroservicio.Domain.Events

Assembly: AlquilerMicroservicio.Domain.dll

Evento de dominio que se lanza cuando un vehículo ha sido alquilado.

```
public class VehicleRentedDomainEvent : INotification
```

Inheritance

object ← VehicleRentedDomainEvent

Implements

INotification

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

VehicleRentedDomainEvent(Guid, Guid, string)

Constructor del evento que asigna los datos del alquiler realizado.

```
public VehicleRentedDomainEvent(Guid rentalId, Guid vehicleId, string customerId)
```

Parameters

`rentalId` Guid

`vehicleId` Guid

`customerId` string

Properties

CustomerId

Identificador del cliente que alquiló el vehículo.

```
public string CustomerId { get; }
```

Property Value

string

RentalId

Identificador del alquiler realizado.

```
public Guid RentalId { get; }
```

Property Value

Guid

VehicleId

Identificador del vehículo alquilado.

```
public Guid VehicleId { get; }
```

Property Value

Guid

Namespace AlquilerMicroservicio.Domain.Interfaces

Interfaces

IRentalRepository

Contrato para el repositorio de alquileres. Define las operaciones básicas sobre la entidad Rental.

IVehicleRepository

Contrato para el repositorio de vehiculos. Define las operaciones permitidas sobre la entidad Vehicle.

Interface IRentalRepository

Namespace: AlquilerMicroservicio.Domain.Interfaces

Assembly: AlquilerMicroservicio.Domain.dll

Contrato para el repositorio de alquileres. Define las operaciones básicas sobre la entidad Rental.

```
public interface IRentalRepository
```

Methods

AddAsync(Rental)

Agrega un nuevo alquiler al sistema de forma asíncrona.

```
Task AddAsync(Rental rental)
```

Parameters

`rental` Rental

Returns

Task

GetActiveRentalByCustomerAsync(string)

Obtiene el alquiler activo de un cliente, si es que tiene uno en curso.

```
Task<Rental?> GetActiveRentalByCustomerAsync(string customerId)
```

Parameters

`customerId` string

Returns

Task<Rental>

GetByIdAsync(Guid)

Busca un alquiler por su identificador.

```
Task<Rental?> GetByIdAsync(Guid id)
```

Parameters

`id` Guid

Returns

Task<Rental>

UpdateAsync(Rental)

Actualiza la informacion de un alquiler existennte.

```
Task UpdateAsync(Rental rental)
```

Parameters

`rental` Rental

Returns

Task

Interface IVehicleRepository

Namespace: AlquilerMicroservicio.Domain.Interfaces

Assembly: AlquilerMicroservicio.Domain.dll

Contrato para el repositorio de vehiculos. Define las operaciones permitidas sobre la entidad Vehicle.

```
public interface IVehicleRepository
```

Methods

AddAsync(Vehicle)

Guarda un nuevo vehículo en el sistema.

```
Task AddAsync(Vehicle vehicle)
```

Parameters

`vehicle` Vehicle

Returns

Task

GetAvailableAsync()

Devuelve la lista de vehiculos que están disponibles para ser alquilados.

```
Task<List<Vehicle>> GetAvailableAsync()
```

Returns

Task<List<Vehicle>>

GetByIdAsync(Guid)

Busca un vehiculo por su identificador.

```
Task<Vehicle?> GetByIdAsync(Guid id)
```

Parameters

`id` Guid

Returns

Task<Vehicle>

UpdateAsync(Vehicle)

Actualiza la información de un vehiculo existente.

```
Task UpdateAsync(Vehicle vehicle)
```

Parameters

`vehicle` Vehicle

Returns

Task

Namespace AlquilerMicroservicio.Domain.Services

Classes

RentalDomainService

Servicio de dominio que gestiona la lógica para alquilar un vehículo. Valida las reglas del negocio y genera el evento correspondiente.

Class RentalDomainService

Namespace: AlquilerMicroservicio.Domain.Services

Assembly: AlquilerMicroservicio.Domain.dll

Servicio de dominio que gestiona la lógica para alquilar un vehículo. Valida las reglas del negocio y genera el evento correspondiente.

```
public class RentalDomainService
```

Inheritance

object ← RentalDomainService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

RentalDomainService(IRentalRepository, IVehicleRepository)

Se inyectan los repositorios necesarios para validar y procesar el alquiler.

```
public RentalDomainService(IRentalRepository rentalRepository,
IVehicleRepository vehicleRepository)
```

Parameters

`rentalRepository` IRentalRepository

`vehicleRepository` IVehicleRepository

Methods

RentVehicleAsync(Guid, Guid, string)

Ejecuta el proceso de alquiler: valida que el cliente no tenga otro activo, verifica que el vehículo exista y esté disponible, y construye el alquiler con su evento.

```
public Task<(Rental Rental, VehicleRentedDomainEvent DomainEvent)> RentVehicleAsync(Guid
rentalId, Guid vehicleId, string customerId)
```

Parameters

`rentalId` Guid

`vehicleId` Guid

`customerId` string

Returns

Task<(Rental Rental, VehicleRentedDomainEvent DomainEvent)>

Namespace AlquilerMicroservicio.Infrastructure

Classes

InfrastructureServiceCollectionExtensions

Clase de extensión que registra los servicios de infraestructura en el contenedor de dependencias.

Class InfrastructureServiceCollectionExtensions

Namespace: AlquilerMicroservicio.Infrastructure

Assembly: AlquilerMicroservicio.Infrastructure.dll

Clase de extensión que registra los servicios de infraestructura en el contenedor de dependencias.

```
public static class InfrastructureServiceCollectionExtensions
```

Inheritance

object ← InfrastructureServiceCollectionExtensions

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

AddInfrastructure(IServiceCollection, IConfiguration)

Registra la configuración de MongoDB, los repositorios, servicios de dominio y el despachador de eventos.

```
public static IServiceCollection AddInfrastructure(this IServiceCollection services,
IConfiguration configuration)
```

Parameters

`services` IServiceCollection

Contenedor de servicios.

`configuration` IConfiguration

Configuración de la aplicación.

Returns

IServiceCollection

El contenedor con los servicios registrados.

Namespace AlquilerMicroservicio.Infrastructure.Events

Classes

DomainEventDispatcher

Implementación del despachador de eventos de dominio que utiliza MediatR para publicar los eventos.

Interfaces

IDomainEventDispatcher

Interfaz para despachar eventos de dominio usando un mecanismo externo.

Class DomainEventDispatcher

Namespace: AlquilerMicroservicio.Infrastructure.Events

Assembly: AlquilerMicroservicio.Infrastructure.dll

Implementación del despachador de eventos de dominio que utiliza MediatR para publicar los eventos.

```
public class DomainEventDispatcher : IDomainEventDispatcher
```

Inheritance

object ← DomainEventDispatcher

Implements

IDomainEventDispatcher

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

DomainEventDispatcher(IMediator)

Se inyecta el mediador para poder publicar los eventos de forma centralizada.

```
public DomainEventDispatcher(IMediator mediator)
```

Parameters

`mediator` IMediator

Methods

DispatchAsync<TEvent>(TEvent, CancellationToken)

Publica el evento usando MediatR, si el evento no es nulo.

```
public Task DispatchAsync<TEvent>(TEvent domainEvent, CancellationToken cancellationToken =
default) where TEvent : notnull
```

Parameters

`domainEvent` TEvent

`cancellationToken` CancellationToken

Returns

Task

Type Parameters

`TEvent`

Interface IDomainEventDispatcher

Namespace: AlquilerMicroservicio.Infrastructure.Events

Assembly: AlquilerMicroservicio.Infrastructure.dll

Interfaz para despachar eventos de dominio usando un mecanismo externo.

```
public interface IDomainEventDispatcher
```

Methods

DispatchAsync<TEvent>(TEvent, CancellationToken)

Lanza un evento de dominio mediante el sistema de publicación.

```
Task DispatchAsync<TEvent>(TEvent domainEvent, CancellationToken cancellationToken =
default) where TEvent : notnull
```

Parameters

`domainEvent` TEvent

`cancellationToken` CancellationToken

Returns

Task

Type Parameters

`TEvent`

Namespace AlquilerMicroservicio.Infrastructure.Repositories

Classes

MongoRentalRepository

Implementación del repositorio de alquileres usando MongoDB como almacenamiento.

MongoVehicleRepository

Implementación del repositorio de vehículos usando MongoDB como base de datos.

Class MongoRentalRepository

Namespace: AlquilerMicroservicio.Infrastructure.Repositories

Assembly: AlquilerMicroservicio.Infrastructure.dll

Implementación del repositorio de alquileres usando MongoDB como almacenamiento.

```
public class MongoRentalRepository : IRentalRepository
```

Inheritance

object ← MongoRentalRepository

Implements

IRentalRepository

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MongoRentalRepository(IMongoDatabase)

Constructor que obtiene la colección de alquileres desde la base de datos MongoDB.

```
public MongoRentalRepository(IMongoDatabase database)
```

Parameters

`database` IMongoDatabase

Methods

AddAsync(Rental)

Inserta un nuevo alquiler en la base de datos.

```
public Task AddAsync(Rental rental)
```

Parameters

`rental` Rental

Returns

Task

GetActiveRentalByCustomerAsync(string)

Obtiene el alquiler activo de un cliente, si tiene alguno en curso.

```
public Task<Rental?> GetActiveRentalByCustomerAsync(string customerId)
```

Parameters

`customerId` string

Returns

Task<Rental>

GetByIdAsync(Guid)

Busca un alquiler por su identificador.

```
public Task<Rental?> GetByIdAsync(Guid id)
```

Parameters

`id` Guid

Returns

Task<Rental>

UpdateAsync(Rental)

Reemplaza la información de un alquiler existente por una versión actualizada.

```
public Task UpdateAsync(Rental rental)
```

Parameters

`rental` Rental

Returns

Task

Class MongoVehicleRepository

Namespace: AlquilerMicroservicio.Infrastructure.Repositories

Assembly: AlquilerMicroservicio.Infrastructure.dll

Implementación del repositorio de vehículos usando MongoDB como base de datos.

```
public class MongoVehicleRepository : IVehicleRepository
```

Inheritance

object ← MongoVehicleRepository

Implements

IVehicleRepository

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MongoVehicleRepository(IMongoDatabase)

Constructor que accede a la colección de vehículos en MongoDB.

```
public MongoVehicleRepository(IMongoDatabase database)
```

Parameters

`database` IMongoDatabase

Methods

AddAsync(Vehicle)

Inserta un vehículo nuevo en la colección.

```
public Task AddAsync(Vehicle vehicle)
```

Parameters

`vehicle` Vehicle

Returns

Task

GetAvailableAsync()

Devuelve una lista con los vehículos que no están alquilados.

```
public Task<List<Vehicle>> GetAvailableAsync()
```

Returns

Task<List<Vehicle>>

GetByIdAsync(Guid)

Busca un vehículo por su identificador.

```
public Task<Vehicle?> GetByIdAsync(Guid id)
```

Parameters

`id` Guid

Returns

Task<Vehicle>

UpdateAsync(Vehicle)

Reemplaza los datos del vehículo con la información nueva.

```
public Task UpdateAsync(Vehicle vehicle)
```

Parameters

`vehicle` Vehicle

Returns

Task