



El futuro digital
es de todos

MinTIC



Formato de Informe de Seguimiento



Universidad de Caldas



Materia: Desarrollo de Aplicaciones Web – Ciclo 4A

Grupo: 12

Docente: Andrés Felipe Escallon Portilla.

Tutora: Yurley Katherine Echeverría Leal.

Sprint 1: Tienda Virtual de Tecnología.

Formato de Informe de Seguimiento

Equipo 2. Techno Team

Integrantes (Nombre completo)	Cédula	Rol	Nivel de participación (Alto, Medio, Bajo, Retirado)
BARRAGAN PLAZAS CARLOS EDUARDO	79536048	Líder de equipo. Administrador de Configuración.	Alto.
BARRAZA RIOS CRISTIAN	1045749373	Tester.	Alto.
BASTIDAS LAME LAURA MARCELA	1061774975	Diseñador UI.	Alto.
CAICEDO BELTRAN JONATHAN	1030579031	Diseñador de Software.	Alto.
CESPEDES RAMIREZ JOSE GIOVANNI	79854497	No se ha presentado.	No se ha presentado.

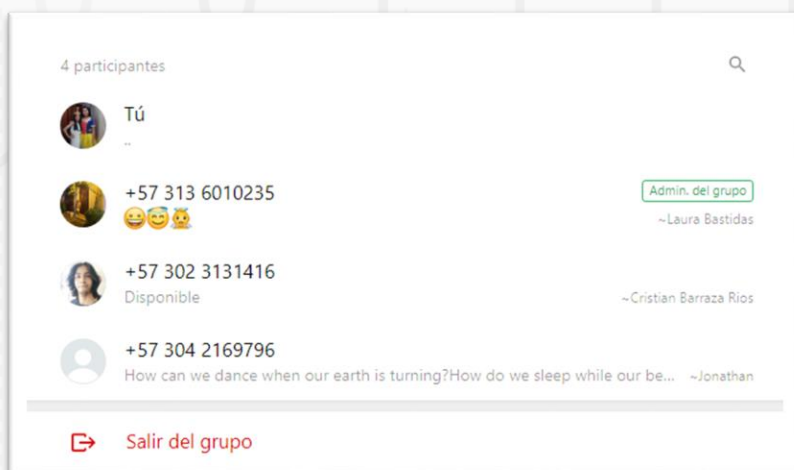


El futuro digital
es de todos

MinTIC

«Mision
TIC2022»

Se realiza la reunión con el equipo, primero nos estábamos comunicando por WhatsApp; donde se creó el grupo:



Como se ha manifestado en el documento se ha tratado de contactar al señor José Giovanni Céspedes Ramírez, pero hasta el momento no ha respondido; nuestra compañera Laura Bastidas lo contacto por celular y mensaje de texto.



Universidad de Caldas

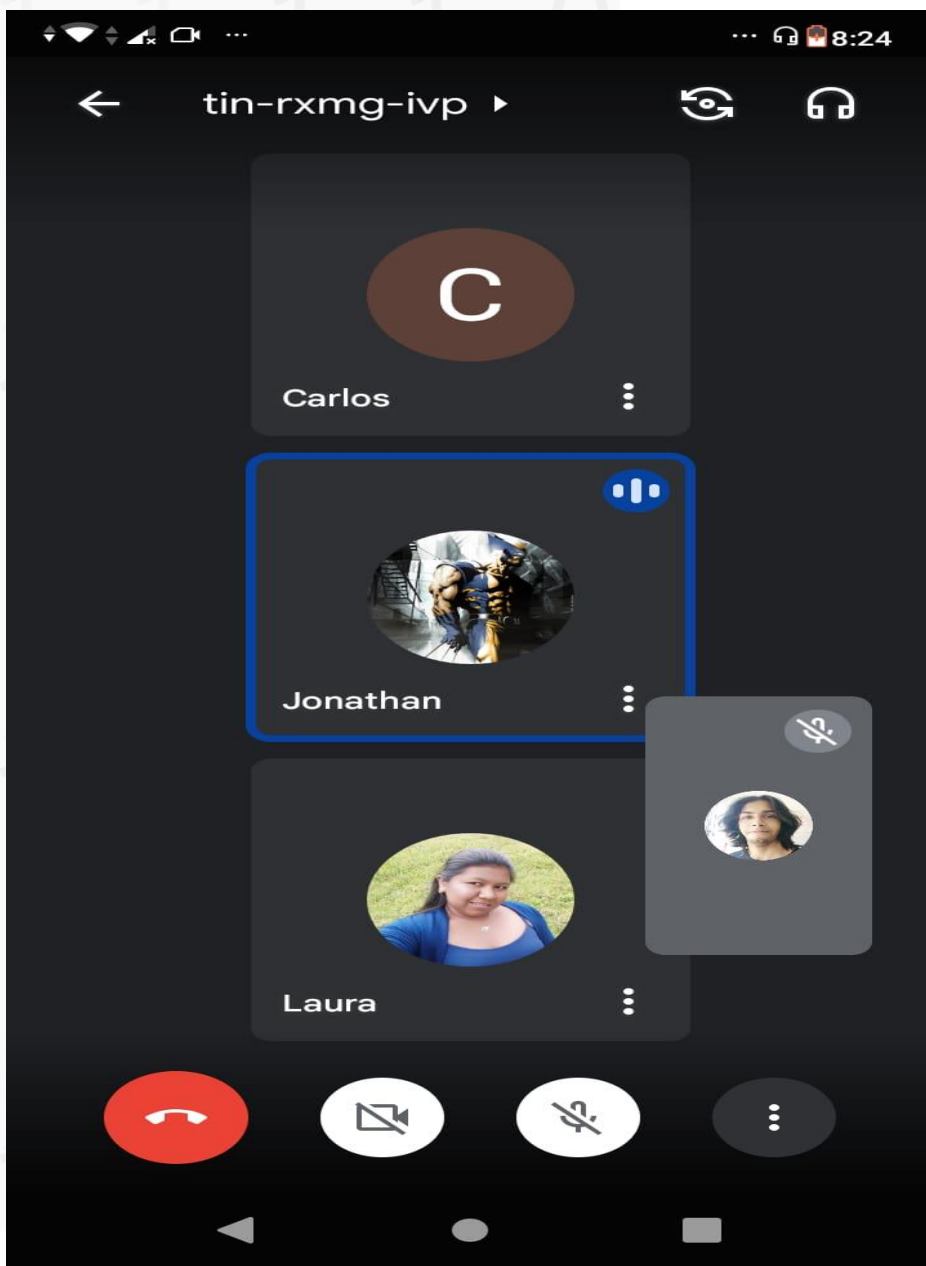


El futuro digital
es de todos

MinTIC

«Misión
TIC 2022»

Se anexa imagen de la reunión por Meet.



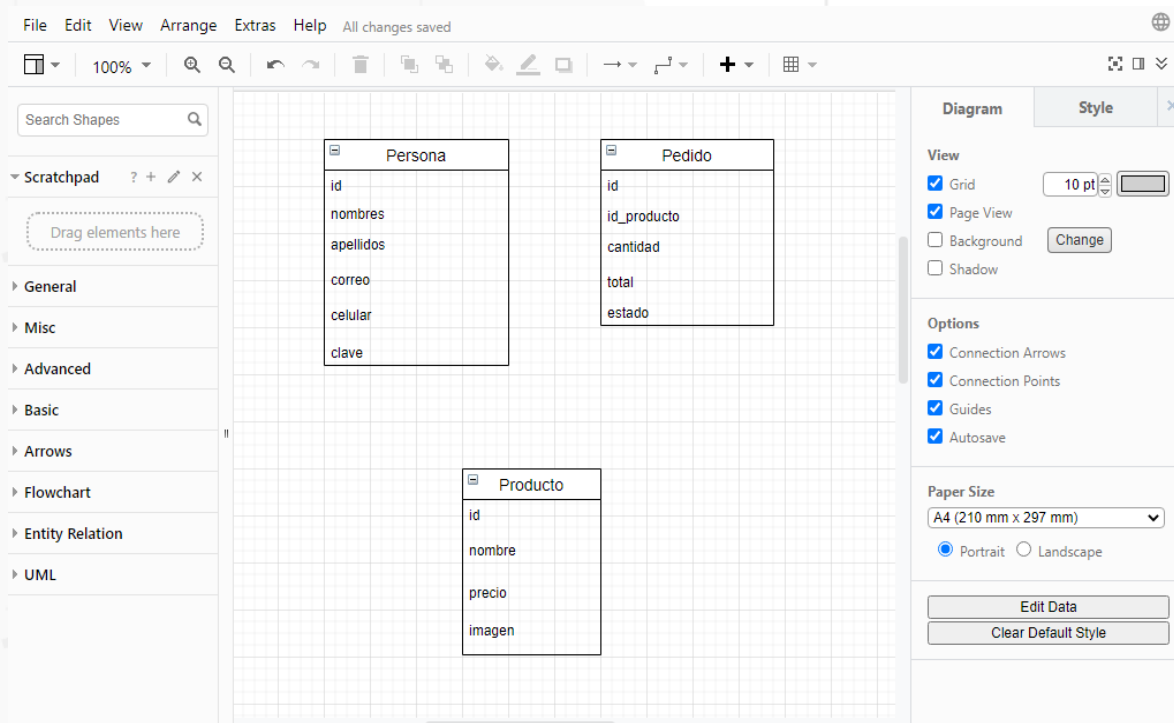
Donde se generaron los roles y la forma en la cual vamos a trabajar.



Universidad de Caldas



Se diseñan las colecciones Persona, Pedido y Producto de la BD PedidosBD, en la aplicación Draw.io



Colecciones	Documentos
Persona	id, nombres, apellidos, correo, celular, clave
Pedido	id, cantidad total estado
Producto	id, nombre precio, imagen

Tenemos las siguientes relaciones:

- una persona puede hacer muchos pedidos.
- un pedido lo hace una persona.
- un producto puede tener muchos pedidos.
- un pedido tiene un producto.

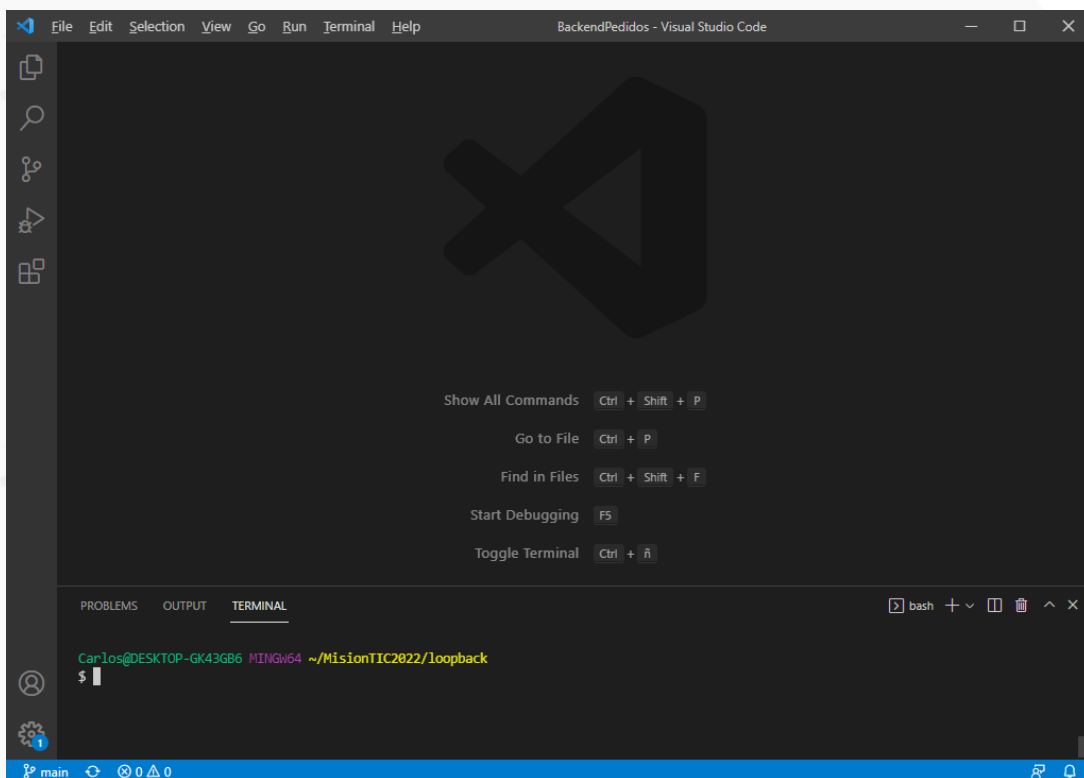


Vamos a crear el proyecto pedidos.

Iniciamos en nuestro editor de código fuente, Visual Studio Code; abrimos una ventana terminal y nos posicionamos en la ruta

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/loopback

Como se observa en la imagen:



Con el comando lb4

Generamos nuestro proyecto con el nombre pedidos.

Queda de la siguiente manera:

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/Loopback

\$ lb4 app

? Nombre de proyectos: pedidos



? Descripción de proyecto: loopback

? Directorio raíz de proyecto: BackendPedidos

? Nombre de clase de aplicación: app

El sistema va a pedir la instalación, de unas extensiones le indicamos al sistema que si esto lo realizamos tecleando Enter.

Como lo podemos visualizar en la siguiente imagen:

```
PROBLEMS OUTPUT TERMINAL
Carlos@DESKTOP-GK43GB6 MINGW64 ~
$ cd MisiónTIC2022

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisiónTIC2022
$ ls
Git/  Loopback/  nodejs/

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisiónTIC2022
$ cd loopback

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisiónTIC2022/loopback
$ lb4 app
? Nombre de proyectos: pedidos
? Descripción de proyecto: loopback
? Directorio raíz de proyecto: BackendPedidos
? Nombre de clase de aplicación: app
? Seleccionar características para habilitarlas en el proyecto Enable eslint, Enable prettier, Enable mocha, Enable loopbackBuild, Enable vscode, Enable docker, Enable repositories, Enable services
```

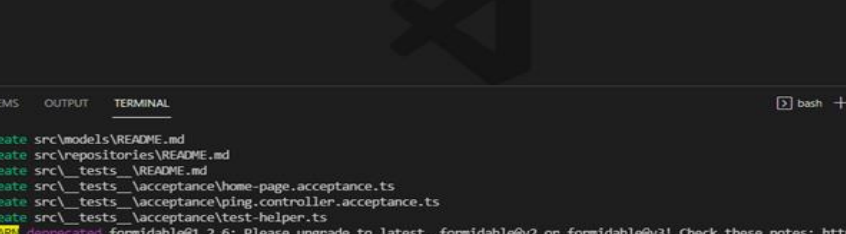


The screenshot shows the Visual Studio Code interface with the 'Terminal' panel active. The terminal displays a list of files to be created for a project named 'BackendPedidos'. The files are listed in a single column, with each line starting with the word 'create' in green. The files include:

- DEVELOPING.md
- package.json
- tsconfig.json
- .vscode\launch.json
- .vscode\settings.json
- .vscode\tasks.json
- .gitignore
- .dockerignore
- Dockerfile
- README.md
- public\index.html
- src\application.ts
- src\index.ts
- src\migrate.ts
- src\openapi-spec.ts
- src\sequence.ts
- src\controllers\index.ts
- src\controllers\ping.controller.ts
- src\controllers\README.md
- src\datasources\README.md
- src\models\README.md
- src\repositories\README.md
- src_tests_README.md
- src_tests_acceptance\home-page.acceptance.ts
- src_tests_acceptance\ping.controller.acceptance.ts
- src_tests_acceptance\test-helper.ts

At the bottom of the terminal, there is a message from npm: `npm WARN deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZEgIAu`. The terminal prompt is `/ reify:@typescript-eslint/parser: timing reifyNode:node_modules/human-signals Completed in 14664ms`.

Una vez que se termina de crear el proyecto, con el comando `npm start` lo inicializamos como se puede visualizar en la imagen.



The screenshot shows the Visual Studio Code interface with the 'TERMINAL' tab active. The terminal output shows the following commands and results:

```
create src\models\README.md
create src\repositories\README.md
create src\_tests\_README.md
create src\_tests\_acceptance\home-page.acceptance.ts
create src\_tests\_acceptance\ping.controller.acceptance.ts
create src\_tests\_acceptance\test-helper.ts
npm WARN deprecated formidable@1.2.6: Please upgrade to latest, formidable@v2 or formidable@v3! Check these notes: https://bit.ly/2ZE
qiaU
npm WARN deprecated uuid@3.4.0: Please upgrade to version 7 or higher. Older versions may use Math.random() in certain circumstance
s, which is known to be problematic. See https://v8.dev/blog/math-random for details.

added 576 packages, and audited 577 packages in 54s

68 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

La aplicación pedidos se ha creado en BackendPedidos

Próximos pasos:

$ cd BackendPedidos
$ npm start

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/loopback
$
```

The status bar at the bottom shows the file 'main' is open, and there are 0 errors, 0 warnings, and 0 info messages. A notification at the bottom right states 'No hay nuevas notificaciones'.



El futuro digital
es de todos

MinTIC

«Mision
TIC 2022»

```
BackendPedidos - Visual Studio Code

EXPLORER
  BACKENDPEDIDOS
    .vscode
    node_modules
    public
    src
    .dockerignore
    .eslintignore
    .eslintrc.js
    .gitignore
    .mocharc.json
    .prettierrc
    .prettierrc
    .yo-rc.json
    DEVELOPING.md
    Dockerfile
    package-lock.json
    package.json
    README.md
    tsconfig.json
    tsconfig.tsbuildinfo

TERMINAL
  bash
  > npm run clean && npm run build
  > pedidos@0.0.1 clean
  > lb-clean dist *.tsbuildinfo .eslintcache
  > pedidos@0.0.1 build
  > lb-tsc
  > pedidos@0.0.1 start
  > node -r source-map-support/register .
  Server is running at http://[::1]:3000
  Try http://[::1]:3000/ping
```

Se Inicia el proyecto:

```
cd BackendPedidos
```

```
$ npm start
```

npm start compila el proyecto.

una vez que termina:

```
> pedidos@0.0.1 start
```

```
> node -r source-map-support/register .
```

Server is running at http://[::1]:3000

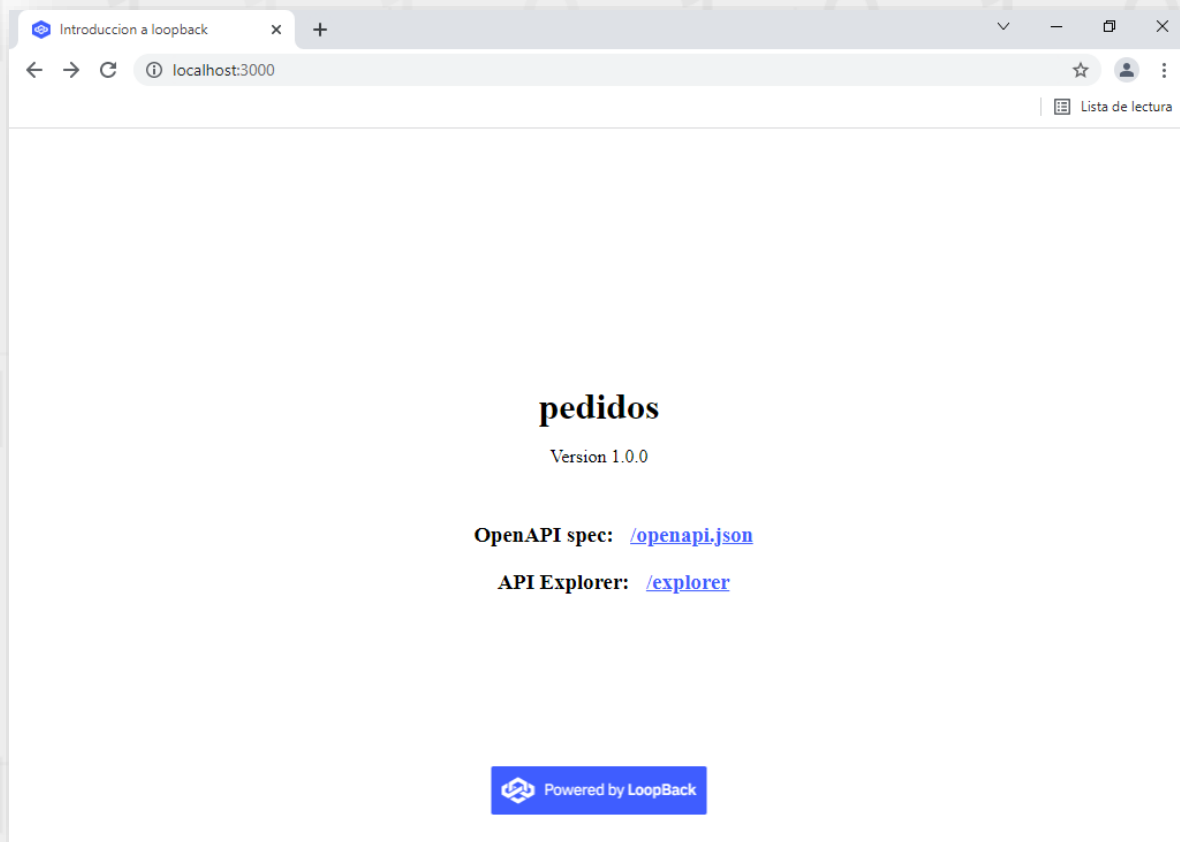
```
Try http://[::1]:3000/ping
```



Universidad de Caldas



En este momento nuestro proyecto pedidos se encuentra ejecutándose y para visualizarlo abrimos el navegador para nuestro caso estamos haciendo uso del navegador Google Chrome y digitamos en la barra de la URL: Localhost:3000 y se nos genera una pantalla como la de la imagen, con esto podemos confirmar que nuestro proyecto pedidos se encuentra correctamente creado y ya se está ejecutando.

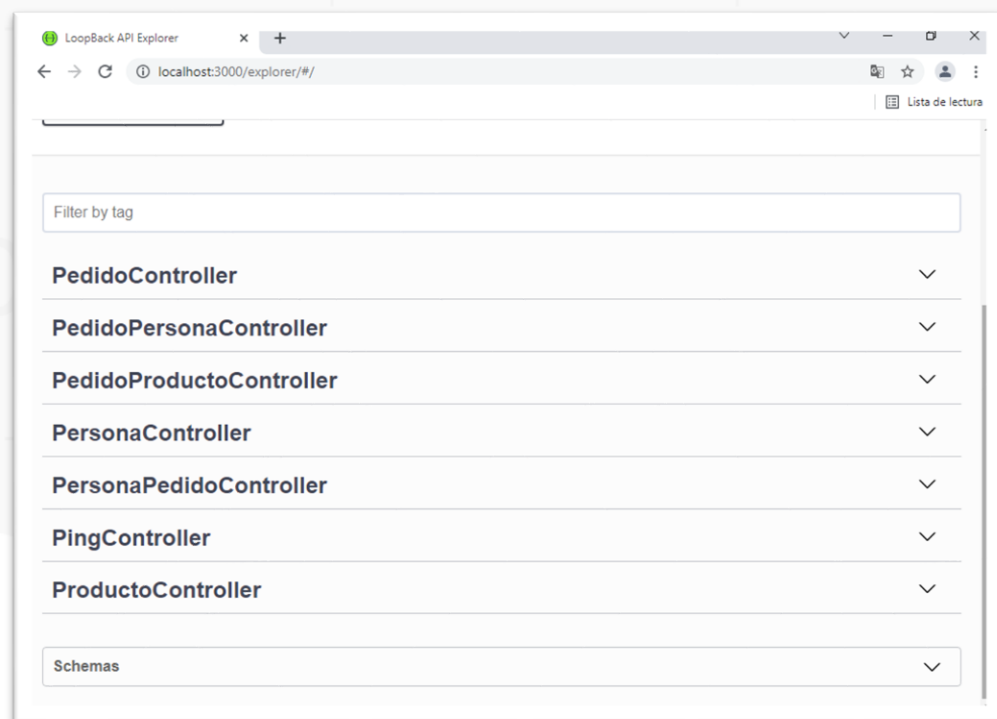
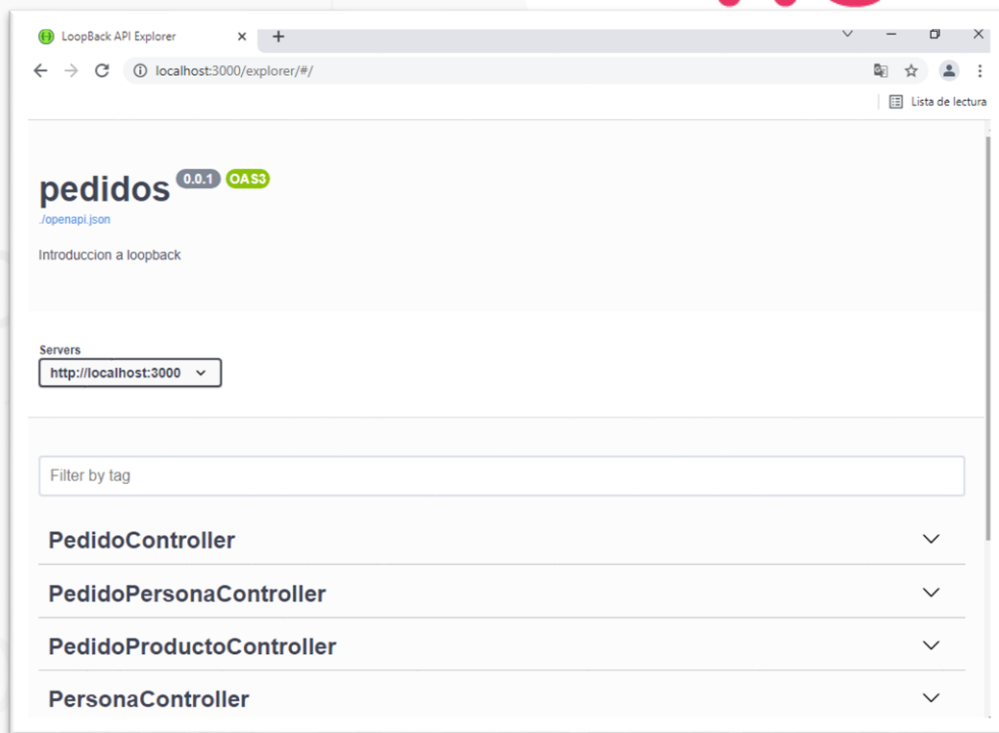




El futuro digital
es de todos

MinTIC

«Mision
TIC 2022»



Universidad de Caldas



El futuro digital
es de todos

MinTIC

«Mision
TIC2022»

```
File Edit Selection View Go Run Terminal Help BackendPedidos - Visual Studio Code

EXPLORER
  BACKENDPEDIDOS
    .vscode
    node_modules
    public
    src
    .dockerignore
    .eslintrc.js
    .gitignore
    .mocharc.json
    .prettierrc
    .yo-rc.json
    DEVELOPING.md
    Dockerfile
    package-lock.json
    package.json
    README.md
    tsconfig.json
    tsconfig.tsbuildinfo

PROBLEMS OUTPUT TERMINAL
  bash + - + x

> npm run clean && npm run build

> pedidos@0.0.1 clean
> lb-clean dist *.tsbuildinfo .eslintcache

> pedidos@0.0.1 build
> lb-tsc

> pedidos@0.0.1 start
> node -r source-map-support/register .

Server is running at http://[::1]:3000
Try http://[::1]:3000/ping

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/Loopback/BackendPedidos (main)
$
```

Para terminar la ejecución del proyecto digitamos las teclas CTRL + C en Visual Studio Code.



Universidad de Caldas



Luego de tener creado el proyecto procedemos a crear el DATASOURCE, MODELOS, REPOSITORIOS y RELACIONES.

Primero el DATASOURCE

Comando: lb4 datasource

nombre: mongodb

En el listado seleccionar para MongoDB

Las características siguientes pueden quedar en blanco y por último se revisa el archivo mongodb.datasource.ts que se encuentra en la carpeta generada datasources.

Luego de creado el archivo se toma la conexión de mongodb atlas y se agrega en el campo url del archivo y se coloca el nombre de la base de datos en la cadena de conexión y la contraseña.

```
src > datasources > mongodb.datasource.ts > config > url
1 import {inject, LifecycleObserver, LifecycleObserver} from '@loopback/core';
2 import {Juggler} from '@loopback/repository';
3
4 const config = {
5   name: 'mongodb',
6   connector: 'mongodb',
7   url: 'mongodb+srv://prog_web:clusterprogweb.1msxm.mongodb.net/PedidosBD?retryWrites=true&w=majority',
8   host: '',
9   port: 0,
10  user: '',
11  password: '',
12  database: '',
13  useNewUrlParser: true
14 };
15
16 // Observe application's life cycle to disconnect the datasource when
17 // application is stopped. This allows the application to be shut down
18 // gracefully. The 'stop()' method is inherited from 'Juggler.DataSource'.
19 // Learn more at https://loopback.io/doc/en/lb4/Life-cycle.html
20 @LifecycleObserver('datasource')
21 export class MongoDBDataSource extends Juggler.DataSource
22   implements LifecycleObserver {
23   static dataSourceName = 'mongodb';
24   static readonly defaultConfig = config;
25
26   constructor(
27     @inject('datasources.config.mongodb', {optional: true})
28     dsConfig: object = config,
29   ) {
30     super(dsConfig);
31   }
32 }
```



A continuación, se generan los MODELOS: Persona, Pedido, Producto

Como ejemplo se colocará la configuración de creación para Persona, aclarando que para generar los otros dos se realiza el mismo proceso.

Comando: lb4 model

nombre: Persona

Tipo de modelo: Entity

Propiedades adicionales: No

Características:

En primer lugar, se define la propiedad id, la cual nos pregunta el tipo, luego confirma que sea la propiedad ID y por último si se incrementa, a lo cual decimos que sí.

Luego una a una se definen las demás colocándoles el nombre y el tipo.

Al finalizar se escribe una propiedad vacía indicando que se terminó de crear el modelo.

id->string->yes->yes

nombres->string->yes

apellidos->string->yes

correo->string->yes

celular->string->yes

clave->string->yes

ENTER

En la carpeta models los encontramos.



El futuro digital
es de todos

MinTIC

«Mision
TIC 2022»

```
src > models > persona.models > ...
1 import {Entity, model, property, hasMany} from '@loopback/repository';
2 import {Pedido} from './pedido.model';
3
4 @model()
5 export class Persona extends Entity {
6   @property({
7     type: 'string',
8     id: true,
9     generated: true,
10  })
11  id?: string;
12
13   @property({
14     type: 'string',
15     required: true,
16  })
17  nombres: string;
18
19   @property({
20     type: 'string',
21     required: true,
22  })
23  apellidos: string;
24
25   @property({
26     type: 'string',
27     required: true,
28  })
29  correo: string;
30
31   @property({
32     type: 'string',
33     required: true,
```

```
src > models > pedido.models > Pedido
1 import {belongsTo, Entity, model, property, hasOne} from '@loopback/repository';
2 import {Persona} from './persona.model';
3 import {Producto} from './producto.model';
4
5 @model()
6 export class Pedido extends Entity {
7   @property({
8     type: 'string',
9     id: true,
10    generated: true,
11  })
12  id?: string;
13
14   @property({
15     type: 'string',
16     required: true,
17  })
18  id_producto: string;
19
20   @property({
21     type: 'number',
22     required: true,
23  })
24  cantidad: number;
25
26   @property({
27     type: 'number',
28     required: true,
29  })
30  total: number;
31
32   @property({
33     type: 'number',
```



Universidad de Caldas



El futuro digital
es de todos

MinTIC

«Mision
TIC 2022»

```
src > models > product.model.ts > ...
1  import {Entity, model, property} from '@loopback/repository';
2
3  @model()
4  export class Producto extends Entity {
5    @property({
6      type: 'string',
7      id: true,
8      generated: true,
9    })
10   id?: string;
11
12   @property({
13     type: 'string',
14     required: true,
15   })
16   nombre: string;
17
18   @property({
19     type: 'number',
20     required: true,
21   })
22   precio: number;
23
24   @property({
25     type: 'string',
26     required: true,
27   })
28   imagen: string;
29
30   @property({
31     type: 'string',
32   })
33   ...
```

De la misma manera creamos los REPOSITORIOS

Comando: `lb4 repository`

MongodbDatasource

Se selecciona con la barra espaciadora Pedido, Persona, Producto

Y por último DefaultCrudRepository



Universidad de Caldas



El futuro digital
es de todos

MinTIC

«Mision
TIC 2022»

The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure. The main editor window shows the `persona.repository.ts` file. The code defines the `PersonaRepository` class, which extends `DefaultCrudRepository`. It includes imports for `inject`, `Getter`, `DefaultCrudRepository`, `repository`, `HasManyRepositoryFactory`, `MongodbDataSource`, `Persona`, `PersonaRelations`, and `Pedido`. The class has a `readonly` property `pedidos` of type `HasManyRepositoryFactory<Pedido, typeof Persona.prototype.id>`. The constructor injects the `MongodbDataSource` and the `PedidoRepository` from the repository. The `constructor` method calls `super(Persona, dataSource)`, `this.createHasManyRepositoryFactoryFor('pedidos', pedidoRepositoryGetter)`, and `this.registerInclusionResolver('pedidos', this.pedidos.inclusionResolver)`.

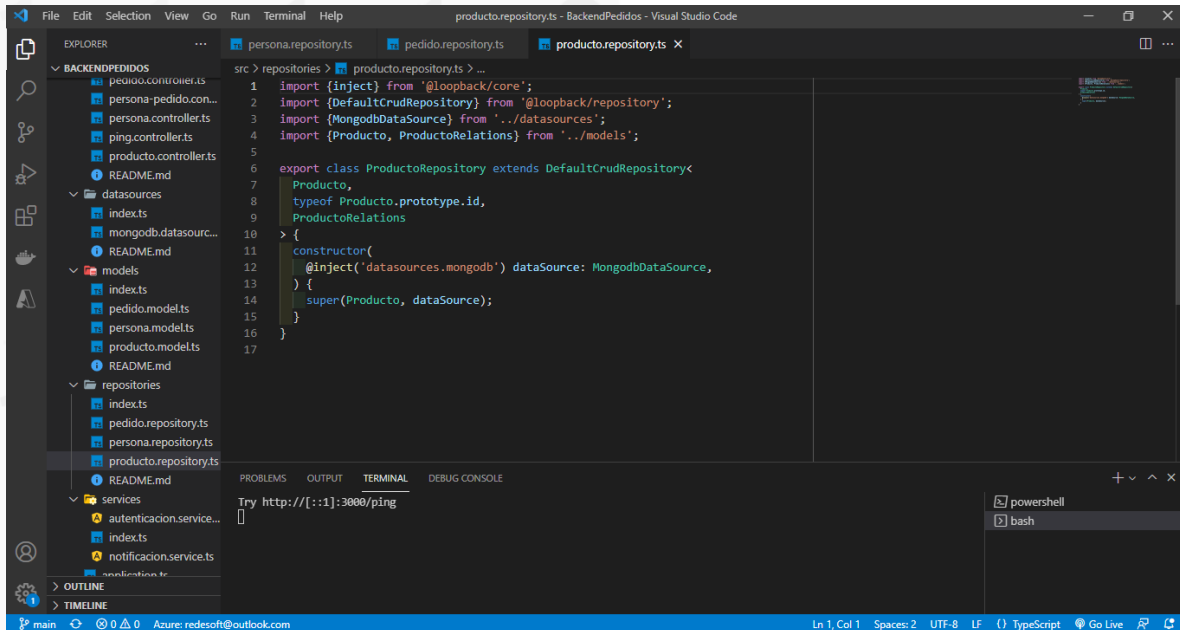
```
src > repositories > persona.repository.ts > ...
1 import {inject, Getter} from '@loopback/core';
2 import {DefaultCrudRepository, repository, HasManyRepositoryFactory} from '@loopback/repository';
3 import {MongodbDataSource} from '../datasources';
4 import {Persona, PersonaRelations, Pedido} from '../models';
5 import {PedidoRepository} from '../pedido.repository';
6
7 export class PersonaRepository extends DefaultCrudRepository<
8   Persona,
9   typeof Persona.prototype.id,
10  PersonaRelations
11 > {
12
13   public readonly pedidos: HasManyRepositoryFactory<Pedido, typeof Persona.prototype.id>;
14
15   constructor(
16     @inject('datasources.mongodb') dataSource: MongodbDataSource, @repository.getter('PedidoRepository') protected pedidoRepositoryGetter: Getter<PedidoRepository>
17   ) {
18     super(Persona, dataSource);
19     this.pedidos = this.createHasManyRepositoryFactoryFor('pedidos', pedidoRepositoryGetter);
20     this.registerInclusionResolver('pedidos', this.pedidos.inclusionResolver);
21   }
22 }
23
```

The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure. The main editor window shows the `pedido.repository.ts` file. The code defines the `PedidoRepository` class, which extends `DefaultCrudRepository`. It includes imports for `inject`, `Getter`, `DefaultCrudRepository`, `repository`, `BelongsToAccessor`, `HasOneRepositoryFactory`, `MongodbDataSource`, `Pedido`, `PedidoRelations`, `Persona`, `Producto`, `PersonaRepository`, and `ProductoRepository`. The class has two `readonly` properties: `persona` of type `BelongsToAccessor<Persona, typeof Pedido.prototype.id>` and `producto` of type `HasOneRepositoryFactory<Producto, typeof Pedido.prototype.id>`. The constructor injects the `MongodbDataSource` and the `PersonaRepository` from the repository. The `constructor` method calls `super(Pedido, dataSource)`, `this.createHasOneRepositoryFactoryFor('producto', productoRepositoryGetter)`, and `this.registerInclusionResolver('producto', this.producto.inclusionResolver)`.

```
src > repositories > pedido.repository.ts > ...
1 import {inject, Getter} from '@loopback/core';
2 import {DefaultCrudRepository, repository, BelongsToAccessor, HasOneRepositoryFactory} from '@loopback/repository';
3 import {MongodbDataSource} from '../datasources';
4 import {Pedido, PedidoRelations, Persona, Producto} from '../models';
5 import {PersonaRepository} from '../persona.repository';
6 import {ProductoRepository} from '../producto.repository';
7
8 export class PedidoRepository extends DefaultCrudRepository<
9   Pedido,
10  typeof Pedido.prototype.id,
11  PedidoRelations
12 > {
13
14   public readonly persona: BelongsToAccessor<Persona, typeof Pedido.prototype.id>;
15
16   public readonly producto: HasOneRepositoryFactory<Producto, typeof Pedido.prototype.id>;
17
18   constructor(
19     @inject('datasources.mongodb') dataSource: MongodbDataSource, @repository.getter('PersonaRepository') protected personaRepositoryGetter: Getter<PersonaRepository>
20   ) {
21     super(Pedido, dataSource);
22     this.producto = this.createHasOneRepositoryFactoryFor('producto', productoRepositoryGetter);
23     this.registerInclusionResolver('producto', this.producto.inclusionResolver);
24   }
25 }
26
```



Universidad de Caldas



Y Ahora creamos las RELACIONES:

Comando: lb4 relation

seleccionar la relación belongsTo

*Pedido

*Persona

*personalId <-Propiedad que relaciona

*persona

*yes

```
@belongsTo(() => Persona)
personaId: string;
```

```
@hasOne(() => Producto)
producto: Producto;
```

```
@hasMany(() => Pedido)
pedidos: Pedido[];
```



Comando: lb4 relation

seleccionar la relación hasMany

*Persona

*Pedido

*personald <-Propiedad que relaciona

*pedidos

*yes

se modifica la entidad de Pedido: producto = id_producto

Comando: lb4 relation

seleccionar la relación hasOne

*Pedido

*Producto

*pedidold <-Propiedad que relaciona

*producto

*yes



Ya terminados estos 4 pilares, seguimos con la creación de los SERVICIOS.

Comando: lb4 service

tipo: Clase de servicio local vinculada a contexto de aplicación

nombre: Notificación

The screenshot shows the Visual Studio Code interface with a project named 'BackendPedidos'. The Explorer sidebar on the left shows the file structure, with 'notification.service.ts' selected under the 'services' folder. The main editor displays the code for 'notification.service.ts':

```
src > services > notification.service.ts > ...
1 import {injectable, /* inject, */ BindingScope} from '@loopback/core';
2
3 @injectable({scope: BindingScope.TRANSIENT})
4 export class NotificacionService {
5   constructor(/* Add @inject to inject parameters */) {}
6
7   /*
8    * Add service methods here
9    */
10 }
11
```

At the bottom, the TERMINAL panel shows a command prompt with the text 'Try http://[::1]:3000/ping'.



Comando: lb4 service

tipo: Clase de servicio local vinculada a contexto de aplicación

nombre: Autenticación

The screenshot shows the Visual Studio Code editor with a project named 'BackendPedidos'. The Explorer sidebar on the left shows the file structure, with the 'services' folder expanded. The main editor displays the file 'autenticacion.service.ts' with the following code:

```
1 import {injectable, /* inject, */ BindingScope} from '@loopback/core';
2
3 @injectable({scope: BindingScope.TRANSIENT})
4 export class AutenticacionService {
5   constructor(/* Add @inject to inject parameters */) {}
6
7   /*
8    * Add service methods here
9    */
10 }
11
```

Below the editor, the TERMINAL panel is open, showing the command 'Try http://[::1]:3000/ping'. The status bar at the bottom indicates the file is at 'Ln 1, Col 1' and the workspace is 'main'.

Como ultima configuración los CONTROLADORES.

Se toma como referencia o ejemplo la creación del controlador para Persona, dado que los demás se realizan de la misma manera con la diferencia de, el nombre de cada controlador y el modelo para el que está asignado.



Comando: lb4 controller

nombre: Persona

especie de controlador: REST con funciones CRUD

nombre del modelo: Persona

nombre del repositorio: PersonaRepository

propiedad: id

tipo: string

omite el ID cuando se crea instancia nueva: Yes

nombre de la vía:/personas

Cabe resaltar que cuando se crean las relaciones, automáticamente se crean los controladores para dichas relaciones.

```
1 import {
2   Count,
3   CountSchema,
4   Filter,
5   FilterExcludingWhere,
6   repository,
7   Where,
8 } from '@loopback/repository';
9 import {
10   post,
11   param,
12   get,
13   getModelSchemaRef,
14   patch,
15   put,
16   del,
17   requestBody,
18   response,
19 } from '@loopback/rest';
20 import {Persona} from '../models';
21 import {PersonaRepository} from '../repositories';
22
23 export class PersonaController {
```



El futuro digital
es de todos

MinTIC

«Mision
TIC 2022»

```
src > controllers > pedido.controller.ts > ...  
22  
23 export class PedidoController {  
24   constructor(  
25     @repository(PedidoRepository)  
26     public pedidoRepository : PedidoRepository,  
27   ) {}  
28  
29   @post('/pedidos')  
30   @response(200, {  
31     description: 'Pedido model instance',  
32     content: {'application/json': {schema: getModelSchemaRef(Pedido)}},  
33   })  
34   async create(  
35     @requestBody({  
36       content: {  
37         'application/json': {  
38           schema: getModelSchemaRef(Pedido, {  
39             title: 'NewPedido',  
40             exclude: ['id'],  
41           }},  
42         },  
43       },  
44     })  
45   )
```

Try http://[::1]:3000/ping

```
src > controllers > producto.controller.ts > ...  
48  
49  
50  
51 @get('/productos/count')  
52 @response(200, {  
53   description: 'Producto model count',  
54   content: {'application/json': {schema: CountsSchema}},  
55 })  
56 async count(  
57   @param.where(Producto) where?: Where<Producto>,  
58 ): Promise<Count> {  
59   return this.productoRepository.count(where);  
60 }  
61  
62 @get('/productos')  
63 @response(200, {  
64   description: 'Array of Producto model instances',  
65   content: {  
66     'application/json': {  
67       schema: {  
68         type: 'array',  
69         items: getModelSchemaRef(Producto, {includeRelations: true}),  
70       },  
71     },  
72   })
```

Try http://[::1]:3000/ping



Universidad de Caldas



El futuro digital
es de todos

MinTIC

‘Mision
TIC 2022’

```
src > controllers > pedido-persona.controller.ts > ...
1  import {
2    repository,
3  } from '@loopback/repository';
4  import {
5    param,
6    get,
7    getModelSchemaRef,
8  } from '@loopback/rest';
9  import {
10    Pedido,
11    Persona,
12  } from '../models';
13  import { PedidoRepository } from '../repositories';
14
15  export class PedidoPersonaController {
16    constructor(
17      @repository(PedidoRepository)
18      public pedidoRepository: PedidoRepository,
19    ) {}
20
21    @get('/pedidos/{id}/persona', {
22      responses: {
23        '200': {
```

Try http://[::1]:3000/ping

```
src > controllers > persona-pedido.controller.ts > ...
25  constructor(
26    @repository(PersonaRepository) protected personaRepository: PersonaRepository,
27  ) {}
28
29  @get('/personas/{id}/pedidos', {
30    responses: {
31      '200': {
32        description: 'Array of Persona has many Pedido',
33        content: {
34          'application/json': {
35            schema: {type: 'array', items: getModelSchemaRef(Pedido)},
36          },
37        },
38      },
39    },
40  })
41  async find(
42    @param.path.string('id') id: string,
43    @param.query.object('filter') filter?: Filter<Pedido>,
44  ): Promise<Pedido[]> {
45    return this.personaRepository.pedidos(id).find(filter);
46  }
47
```

Try http://[::1]:3000/ping



Universidad de Caldas



El futuro digital
es de todos

MinTIC

«Mision
TIC2022»

INICIAR EL SERVIDOR

Comando: npm start

ingresamos al navegador

OpenAPI

Explorer

Como se visualiza en la imagen.

```
File Edit Selection View Go Run Terminal Help BackendPedidos - Visual Studio Code

EXPLORER
  BACKENDPEDIDOS
    .vscode
    node_modules
    public
    src
    _tests_
    controllers
    datasources
    TS index.ts
    TS mongodb.datasource.ts
    README.md
    models
    repositories
    services
    TS application.ts
    TS index.ts
    TS migrate.ts
    TS openapi-spec.ts
    TS sequence.ts
    .dockerignore
    .eslintignore
    .eslintrc.js
    .gitignore
    .mocharc.json
    .prettierrc
    .yo-rc.json
    OUTLINE
    TIMELINE

PROBLEMS OUTPUT TERMINAL
  bash + - + x

Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/Loopback/BackendPedidos (main)
$ npm start

> pedidos@0.0.1 prestart
> npm run rebuild

> pedidos@0.0.1 rebuild
> npm run clean && npm run build

> pedidos@0.0.1 clean
> lb-clean dist *.tsbuildinfo .eslintcache

> pedidos@0.0.1 build
> lb-tsc

> pedidos@0.0.1 start
> node -r source-map-support/register .

Server is running at http://[::1]:3000
Try http://[::1]:3000/ping
```



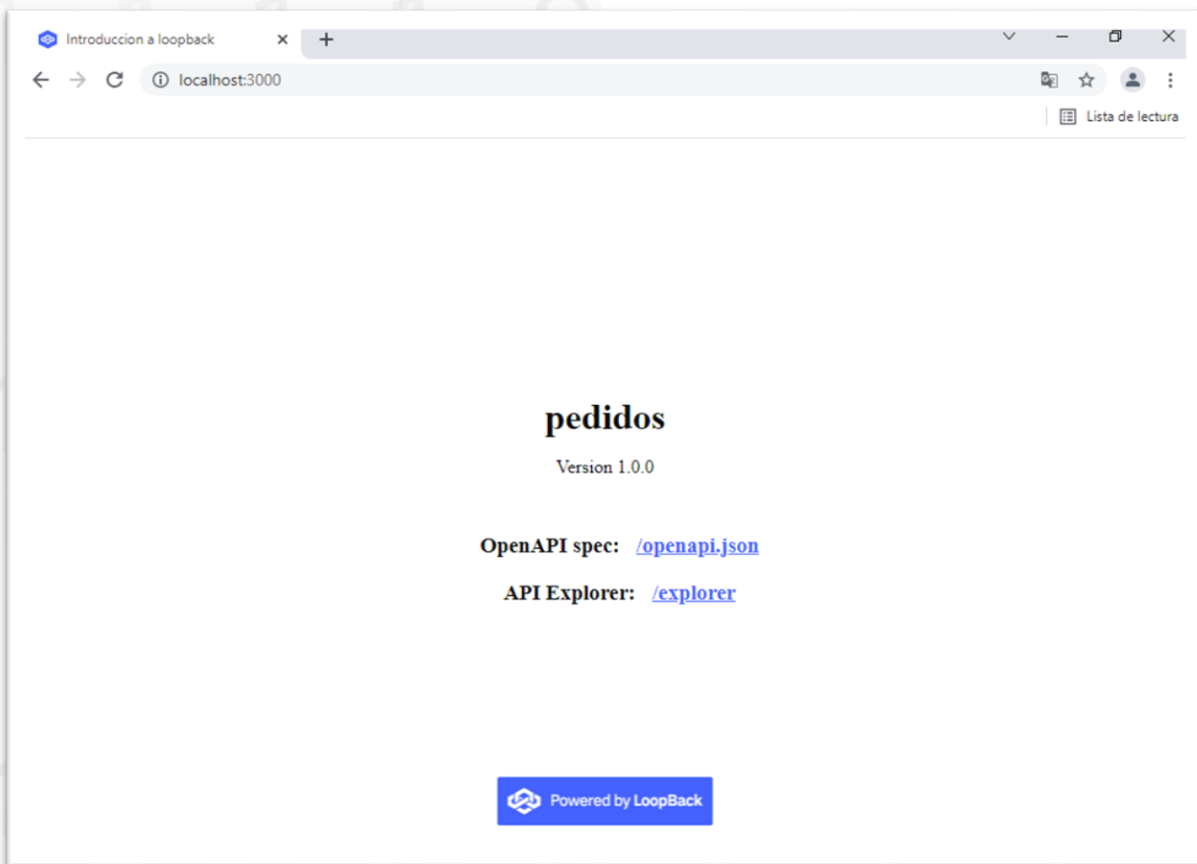
Universidad de Caldas



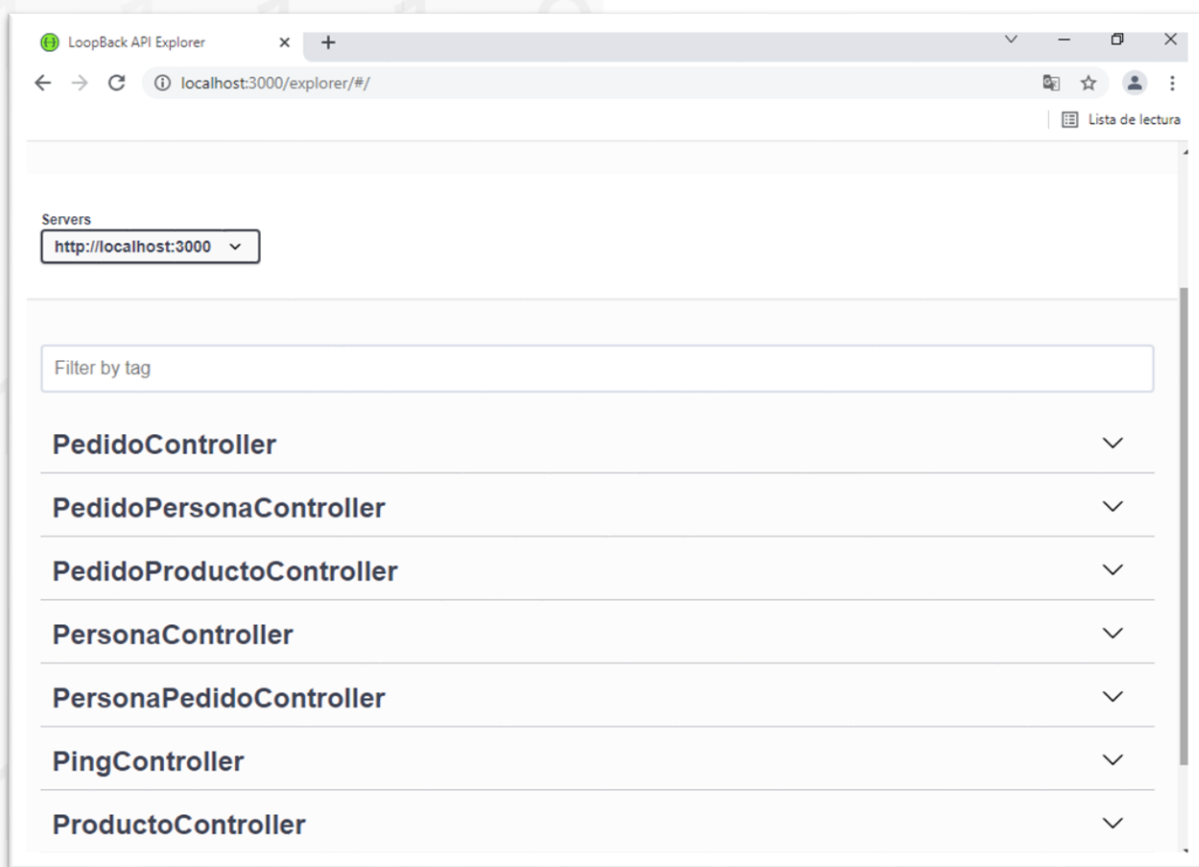
El futuro digital
es de todos

MinTIC

‘Mision
TIC2022’

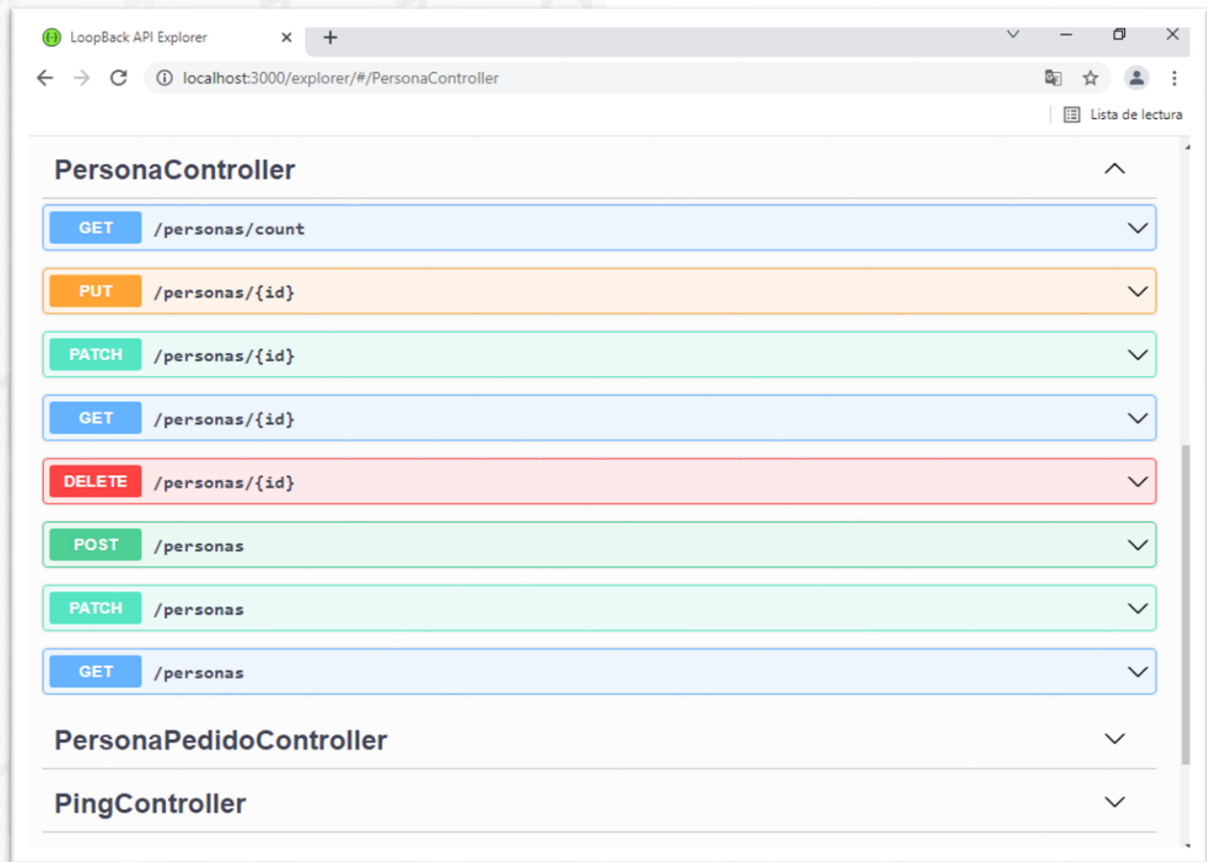


Universidad de Caldas



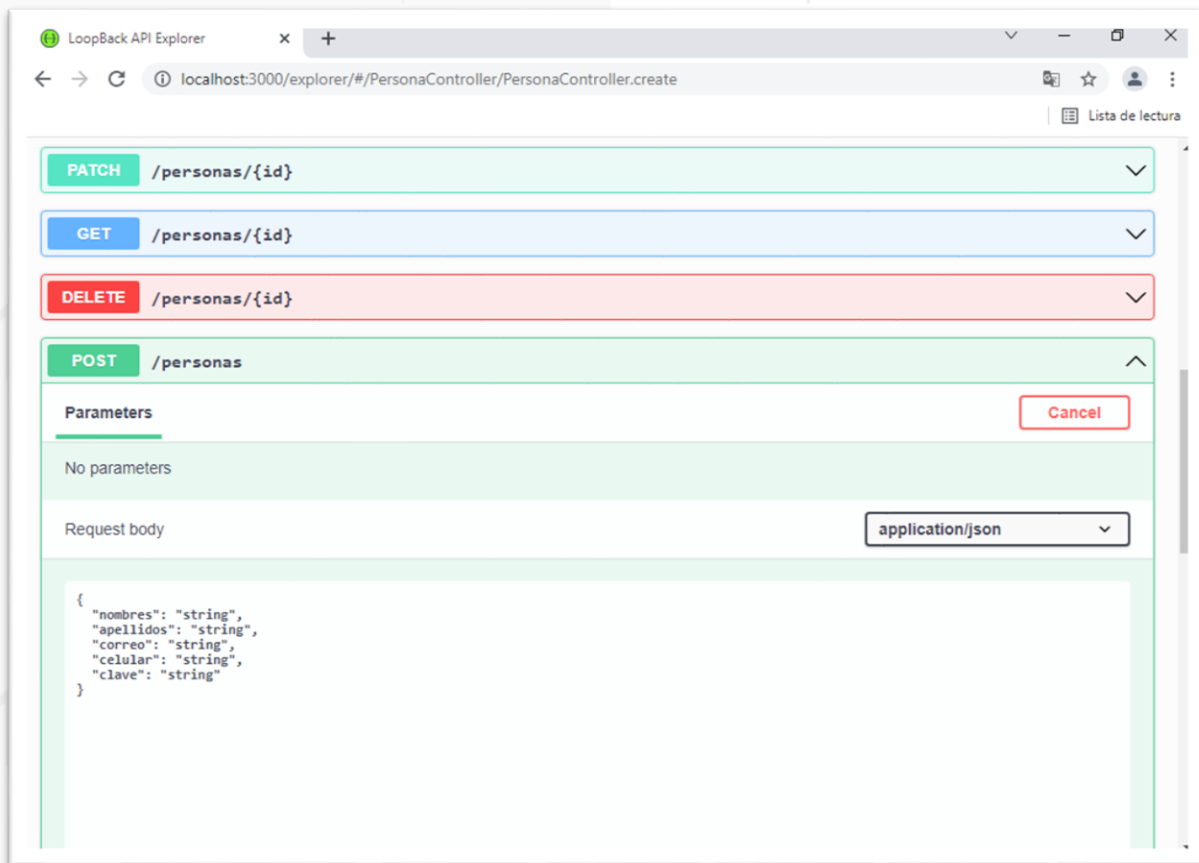
Vamos a realizar una prueba ingresando los datos de una persona
seleccionamos la opción **PersonaController**

Como se visualiza en la imagen.



Con la opción POST, se despliega el menú, se ingresan los datos de la persona para nuestro ejemplo:

```
{  
  "nombres": "string",  
  "apellidos": "string",  
  "correo": "string",  
  "celular": "string",  
  "clave": "string"  
}
```

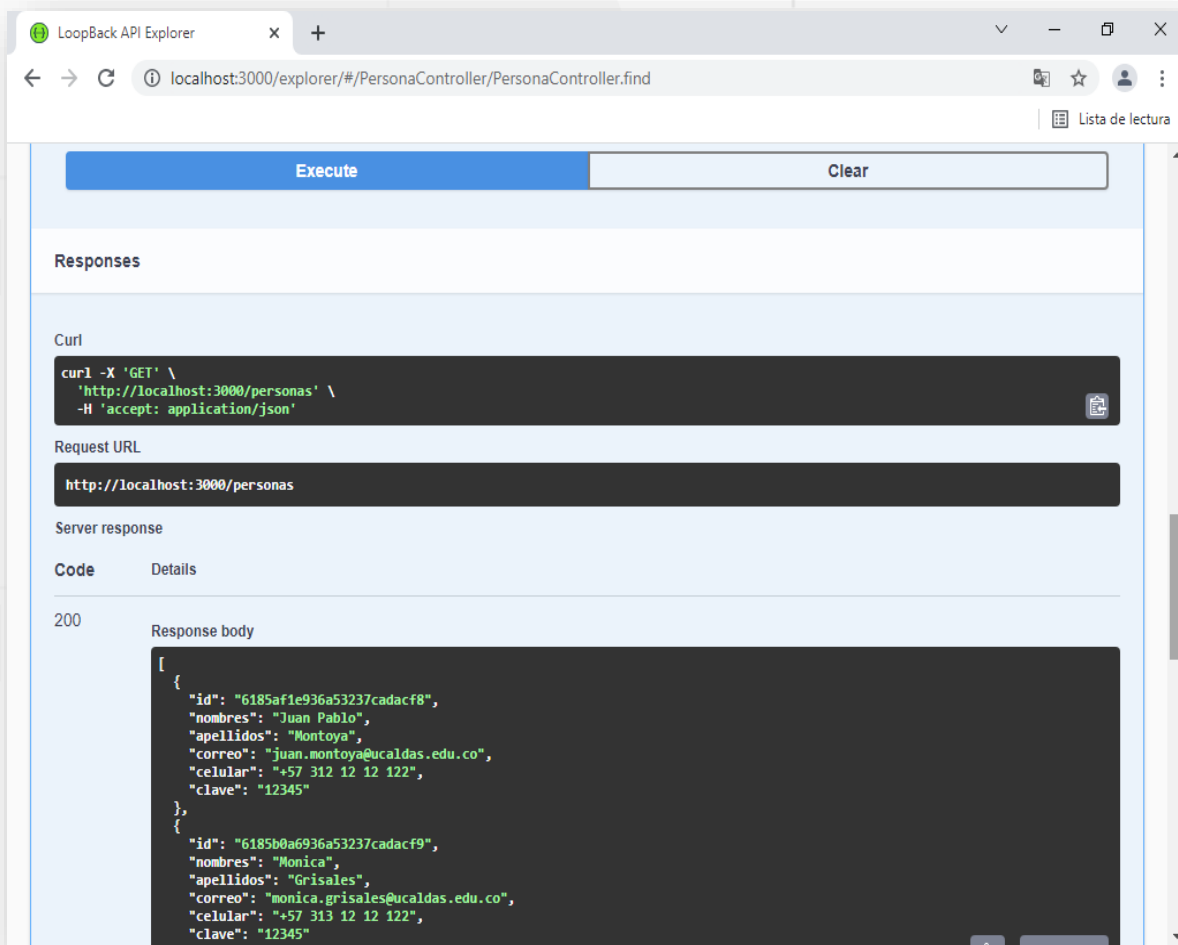


Se da click en la opción execute.

Con la opción GET verificamos que los datos ingresados ya se encuentran grabados.



Como se visualiza en la imagen.



Como podemos observar el proyecto se encuentra funcionando, ahora procedemos a guardar el código en el repositorio.

Iniciamos con el comando `Git init`.



El futuro digital
es de todos

MinTIC

«Mision
TIC2022»

```
File Edit Selection View Go Run Terminal Help BackendPedidos - Visual Studio Code

EXPLORER
  BACKENDPEDIDOS
    .vscode
    node_modules
    public
    src
    .dockerignore
    .eslintrc
    .gitignore
    .mocharc.json
    .prettierrc
    .yo-rc.json
    DEVELOPING.md
    Dockerfile
    package-lock.json
    package.json
    README.md
    tsconfig.json
    tsconfig.tsbuildinfo

PROBLEMS OUTPUT TERMINAL
  bash
  The file will have its original line endings in your working directory
  warning: LF will be replaced by CRLF in src/sequence.ts.
  The file will have its original line endings in your working directory
  warning: LF will be replaced by CRLF in src/services/autenticacion.service.ts.
  The file will have its original line endings in your working directory
  warning: LF will be replaced by CRLF in src/services/index.ts.
  The file will have its original line endings in your working directory
  warning: LF will be replaced by CRLF in src/services/notificacion.service.ts.
  The file will have its original line endings in your working directory
  warning: LF will be replaced by CRLF in tsconfig.json.
  The file will have its original line endings in your working directory
  Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/Loopback/BackendPedidos (master)
  $
```

```
File Edit Selection View Go Run Terminal Help BackendPedidos - Visual Studio Code

EXPLORER
  BACKENDPEDIDOS
    .vscode
    node_modules
    public
    src
    .dockerignore
    .eslintrc
    .gitignore
    .mocharc.json
    .prettierrc
    .yo-rc.json
    DEVELOPING.md
    Dockerfile
    package-lock.json
    package.json
    README.md
    tsconfig.json
    tsconfig.tsbuildinfo

PROBLEMS OUTPUT TERMINAL
  bash
  create mode 100644 src/repositories/index.ts
  create mode 100644 src/repositories/pedido.repository.ts
  create mode 100644 src/repositories/persona.repository.ts
  create mode 100644 src/repositories/producto.repository.ts
  create mode 100644 src/sequence.ts
  create mode 100644 src/services/autenticacion.service.ts
  create mode 100644 src/services/index.ts
  create mode 100644 src/services/notificacion.service.ts
  create mode 100644 tsconfig.json
  Carlos@DESKTOP-GK43GB6 MINGW64 ~/MisionTIC2022/Loopback/BackendPedidos (master)
  $
```



Universidad de Caldas



El futuro digital
es de todos

MinTIC

«Mision
TIC2022»

WhatsApp x DAW-C4A-12: Semana x LoopBack API Explorer x prowebmisiontic/app- x +

github.com/prowebmisiontic/app-pedidos-backend

main 1 branch 0 tags Go to file Add file Code

About
Backend para la aplicación de pedidos.
Readme

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages

TypeScript	92.9%	HTML	5.0%
Dockerfile	1.9%	JavaScript	0.2%

File	Version	Time
.vscode	versión 1.0.0 de la aplicación pedidos	33 minutes ago
public	versión 1.0.0 de la aplicación pedidos	33 minutes ago
src	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.dockerignore	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.eslintignore	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.eslintrc.js	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.gitignore	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.mocha.js	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.prettierrc	versión 1.0.0 de la aplicación pedidos	33 minutes ago
.yo-rc.json	versión 1.0.0 de la aplicación pedidos	33 minutes ago
DEVELOPING.md	versión 1.0.0 de la aplicación pedidos	33 minutes ago
Dockerfile	versión 1.0.0 de la aplicación pedidos	33 minutes ago
README.md	versión 1.0.0 de la aplicación pedidos	33 minutes ago
package-lock.json	versión 1.0.0 de la aplicación pedidos	33 minutes ago

Las líneas de comando que se utilizan:

Carlos@DESKTOP-GK43GB6MINGW64
~/MisionTIC2022/Loopback/BackendPedidos

\$ git init

Initialized empty Git repository in
C:/Users/Carlos/MisionTIC2022/Loopback/BackendPedidos/.git/



Universidad de Caldas



```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos (master)
```

```
$
```

una vez se inicia donde se encuentran las instrucciones
se torna de color verde una letra U nuestros archivos nuevos

Vamos a agregar toda nuestra información al área de preparación
esto lo hacemos con el comando
git add .

se agregan todos los archivos que estén pendientes de ser agregados

```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos
```

```
$ git init
```

Initialized empty Git repository in
C:/Users/Carlos/MisionTIC2022/Loopback/BackendPedidos/.git/

```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos (master)
```

```
$ git add .
```

warning: LF will be replaced by CRLF in .dockerignore.



Una vez se termina la ejecución del comando podemos visualizar que la letra U fue reemplazada por la letra A.

Podemos ver como están los archivos con git status.

Vamos a crear un repositorio nuevo con el nombre app-pedidos-backend

Recuerde que los comandos que utilizamos:

git init Ya lo utilizamos.

git add Ya lo utilizamos.

git commit -m "first commit" Ya lo utilizamos.

git branch -M main Iniciamos desde este comando.

git remote add origin <https://github.com/prowebmisiontic/app-pedidos-backend.git>

git push -u origin main

Carlos@DESKTOP-GK43GB6 MINGW64
~/MisionTIC2022/Loopback/BackendPedidos (master)

\$ git branch -M main



```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos (main)
```

```
$ git remote add origin https://github.com/prowebmisiontic/app-  
pedidos-backend.git
```

```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos (main)
```

```
$
```

Finalmente vamos a subir el código a nuestro Repositorio.

Se utiliza el ultimo comando

```
git push -u origin main
```

Una vez que se ejecute el sistema genera el siguiente mensaje:

```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos (master)
```

```
$ git branch -M main
```

```
Carlos@DESKTOP-GK43GB6 MINGW64  
~/MisionTIC2022/Loopback/BackendPedidos (main)
```

```
$ git remote add origin https://github.com/prowebmisiontic/app-  
pedidos-backend.git
```



```
Carlos@DESKTOP-GK43GB6 MINGW64
~/MisionTIC2022/Loopback/BackendPedidos (main)

$ git push -u origin main

Enumerating objects: 64, done.
Counting objects: 100% (64/64), done.
Delta compression using up to 2 threads
Compressing objects: 100% (59/59), done.
Writing objects: 100% (64/64), 132.54 KiB | 2.76 MiB/s, done.
Total 64 (delta 9), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (9/9), done.
To https://github.com/prowebmisiontic/app-pedidos-backend.git
* [new branch]    main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

```
Carlos@DESKTOP-GK43GB6 MINGW64
~/MisionTIC2022/Loopback/BackendPedidos (main)

$

ya hemos terminado, pero podemos verificar en GitHub.
```

<https://github.com/prowebmisiontic/app-pedidos-backend>

Cuenta en GitHub.

User: prowebmisiontic

Password: ProgWebMintic2022