

UNIVERSIDAD PRIVADA DE TACNA



INGENIERIA DE SISTEMAS

TEMA:

**Proyecto Final de Unidad**

**CURSO:**

BASE DE DATOS II

**DOCENTE(ING):**

Patrick Jose Cuadros Quiroga

Integrantes:

Wilfredo Vilca Chambilla

(2006028540)

# Índice

1. PROBLEMA	1
2. MARCO TEORICO	2
3. DESARROLLO	5

# 1. PROBLEMA

La Empresa actualmente no cuenta con tecnología que esté orientada a guardar información de la empresa; esta es una empresa que comenzó hace poco (7 meses aproximadamente), sus primeros procesos eran netamente manuales, pero ahora necesitan por ejemplo adquisición de un Sistema de Inventario, con el cual se puede tener la información con la que se trabaja, más ordenada y más segura al igual que el control de ventas, control de empleados, etc.

La empresa en cuestión requiere un sistema de información el cual le permita controlar los datos correspondientes a la empresa como: datos de materiales, datos de ventas, datos de empleados, datos de inventariado, etc.



## 2. MARCO TEORICO

### ¿Qué es Microsoft Visual Studio?

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación, tales como C++, C sharp, Visual Basic .NET, F sharp, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y consolas, entre otros.

### ¿Qué es Microsoft SQL Server?

Microsoft SQL Server es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio) es Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).

Dentro de los competidores más destacados de SQL Server están: Oracle, MariaDB, MySQL, PostgreSQL. SQL Server ha estado tradicionalmente disponible solo para sistemas operativos Windows de Microsoft, pero desde 2016 está disponible para GNU/Linux, y a partir de 2017 para Docker también.

Puede ser configurado para utilizar varias instancias en el mismo servidor físico, la primera instalación lleva generalmente el nombre del servidor, y las siguientes - nombres específicos (con un guion invertido entre el nombre del servidor y el nombre de la instalación).

### ¿Qué es un Diagrama de Caso de Uso?

En el Lenguaje de Modelado Unificado, un diagrama de casos de uso es una forma de diagrama de comportamiento UML mejorado. El Lenguaje de Modelado Unificado (UML), define una notación gráfica para representar casos de uso llamada modelo de casos de uso. UML no define estándares para que el formato escrito describa los casos de uso, y así mucha gente no entiende que esta notación gráfica define la naturaleza de un caso de uso; sin embargo, una notación gráfica puede solo dar una vista general simple de un caso de uso o un conjunto de casos de uso. Los diagramas de casos de uso son a menudo confundidos con los casos de uso. Mientras los dos conceptos están relacionados, los casos de uso son mucho más detallados que los diagramas de casos de uso. En los conceptos se debe detallar más de un caso de uso para poder identificar qué es lo que hace un caso de uso.

- La descripción escrita del comportamiento del sistema al afrontar una tarea de negocio o un requisito de negocio. Esta descripción se enfoca en el valor suministrado por el sistema a entidades externas tales como usuarios humanos u otros sistemas.

- La posición o contexto del caso de uso entre otros casos de uso. Dado que es un mecanismo de organización, un conjunto de casos de uso coherentes y consistentes promueven una imagen fácil de comprender del comportamiento del sistema, un entendimiento común entre el cliente/propietario/usuario y el equipo de desarrollo.

### **¿Qué es un Diagrama de Clases?**

En ingeniería de software, un diagrama de clases en Lenguaje Unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

### **¿Qué es un Diagrama de Entidad – Relación?**

Un diagrama entidad-relación, también conocido como modelo entidad relación o ERD, es un tipo de diagrama de flujo que ilustra cómo las "entidades", como personas, objetos o conceptos, se relacionan entre sí dentro de un sistema. Los diagramas ER se usan a menudo para diseñar o depurar bases de datos relacionales en los campos de ingeniería de software, sistemas de información empresarial, educación e investigación. También conocidos como los ERD o modelos ER, emplean un conjunto definido de símbolos, tales como rectángulos, diamantes, óvalos y líneas de conexión para representar la interconexión de entidades, relaciones y sus atributos. Son un reflejo de la estructura gramatical y emplean entidades como sustantivos y relaciones como verbos.

### **¿Qué es ORM?**

Un ORM es un modelo de programación que permite mapear las estructuras de una base de datos relacional (SQL Server, Oracle, MySQL, etc.), en adelante RDBMS (Relational Database Management System), sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones. Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas o base de datos virtual definida en el ORM, de tal modo que las acciones CRUD (Create, Read, Update, Delete) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del ORM.

### **¿Qué es Entity Framework?**

Es una tecnología desarrollada por Microsoft, que a través de ADO.NET genera un conjunto de objetos que están directamente ligados a una Base de Datos, permitiendo a los desarrolladores manejar dichos objetos en lugar de utilizar lenguaje SQL contra la Base de Datos.

### **¿Qué son Pruebas Unitarias?**

En programación, una prueba unitaria es una forma de comprobar el correcto funcionamiento de una unidad de código. Por ejemplo, en diseño estructurado o en diseño funcional una función o un procedimiento, en diseño orientado a objetos una clase. Esto sirve para asegurar que cada unidad funcione correctamente y eficientemente por separado. Además de verificar que el código hace lo que tiene que hacer, verificamos que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, que, si el estado inicial es válido, entonces el estado final es válido también.

La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto. Luego, con las Pruebas de Integración, se podrá

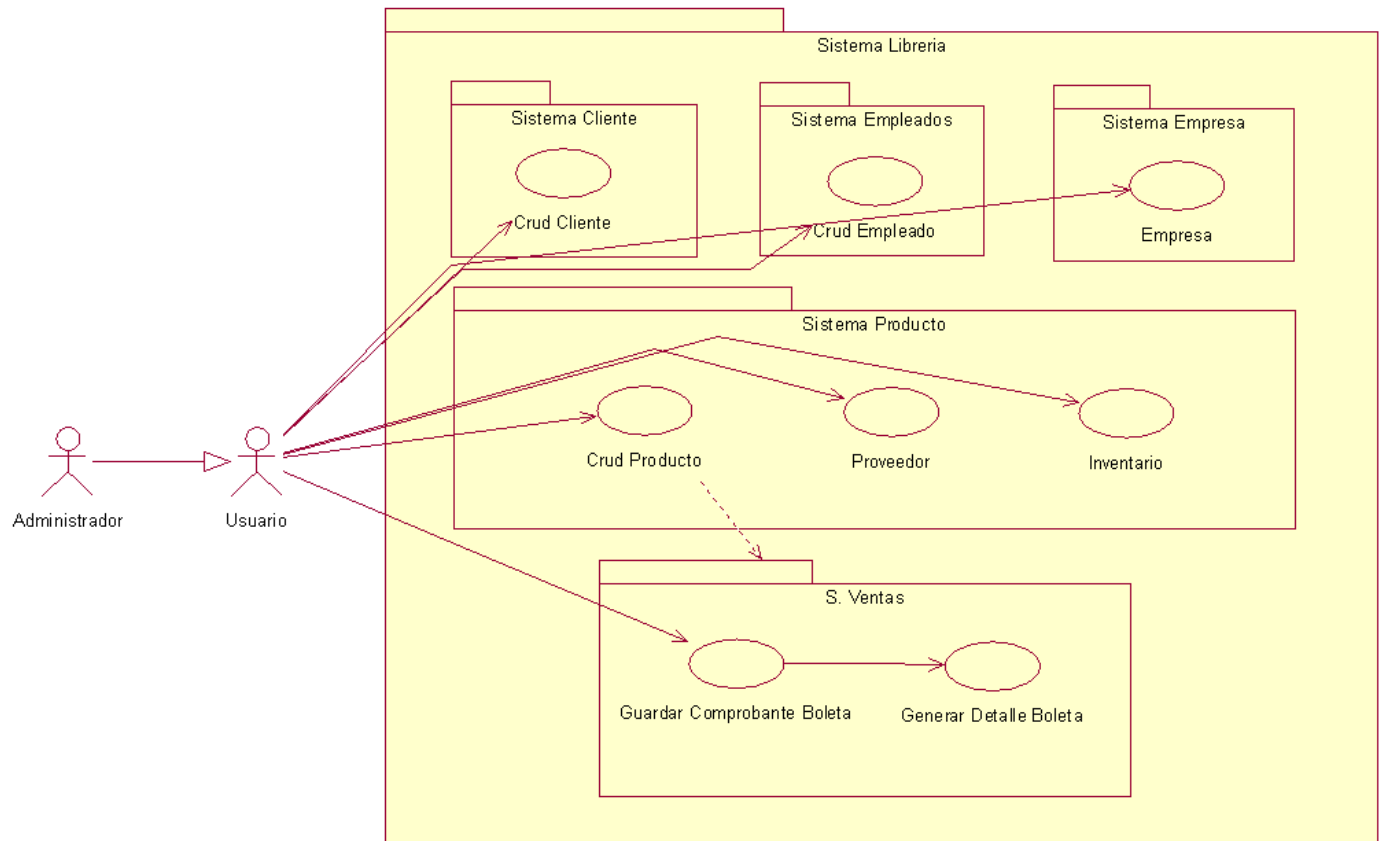
asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

### **¿Qué es un Api EndPoint?**

Una interfaz de programación de aplicaciones (API) permite que dos sistemas se comuniquen entre sí. Una API esencialmente proporciona el lenguaje y el contrato de cómo interactúan los dos sistemas. Cada API tiene documentación y especificaciones que determinan cómo se puede transferir la información. Al igual que se representa una página web, las API pueden usar solicitudes HTTP para obtener información de una aplicación web o servidor web.

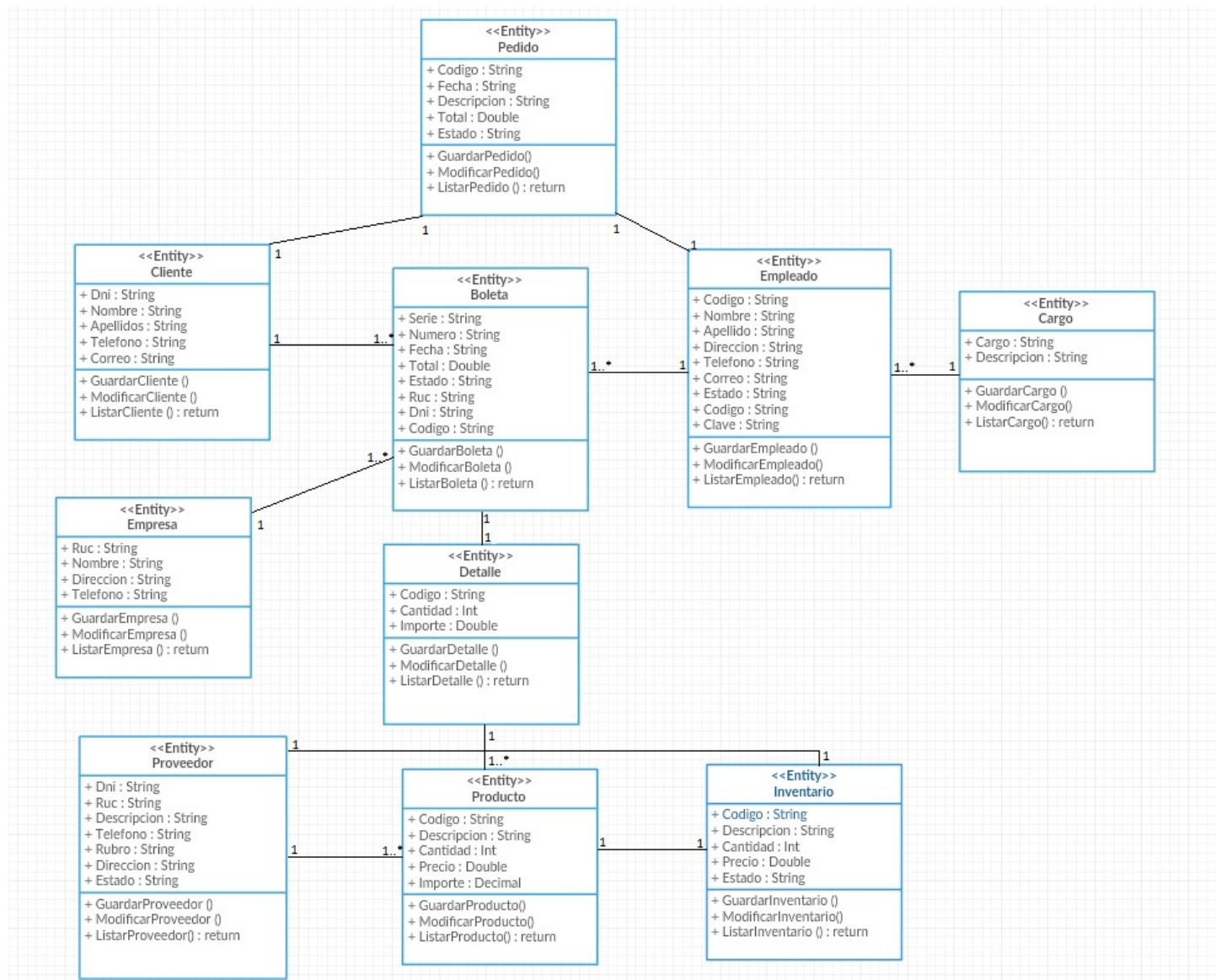
## 3. DESARROLLO

### 3.1 ANALISIS (Casos de Uso)



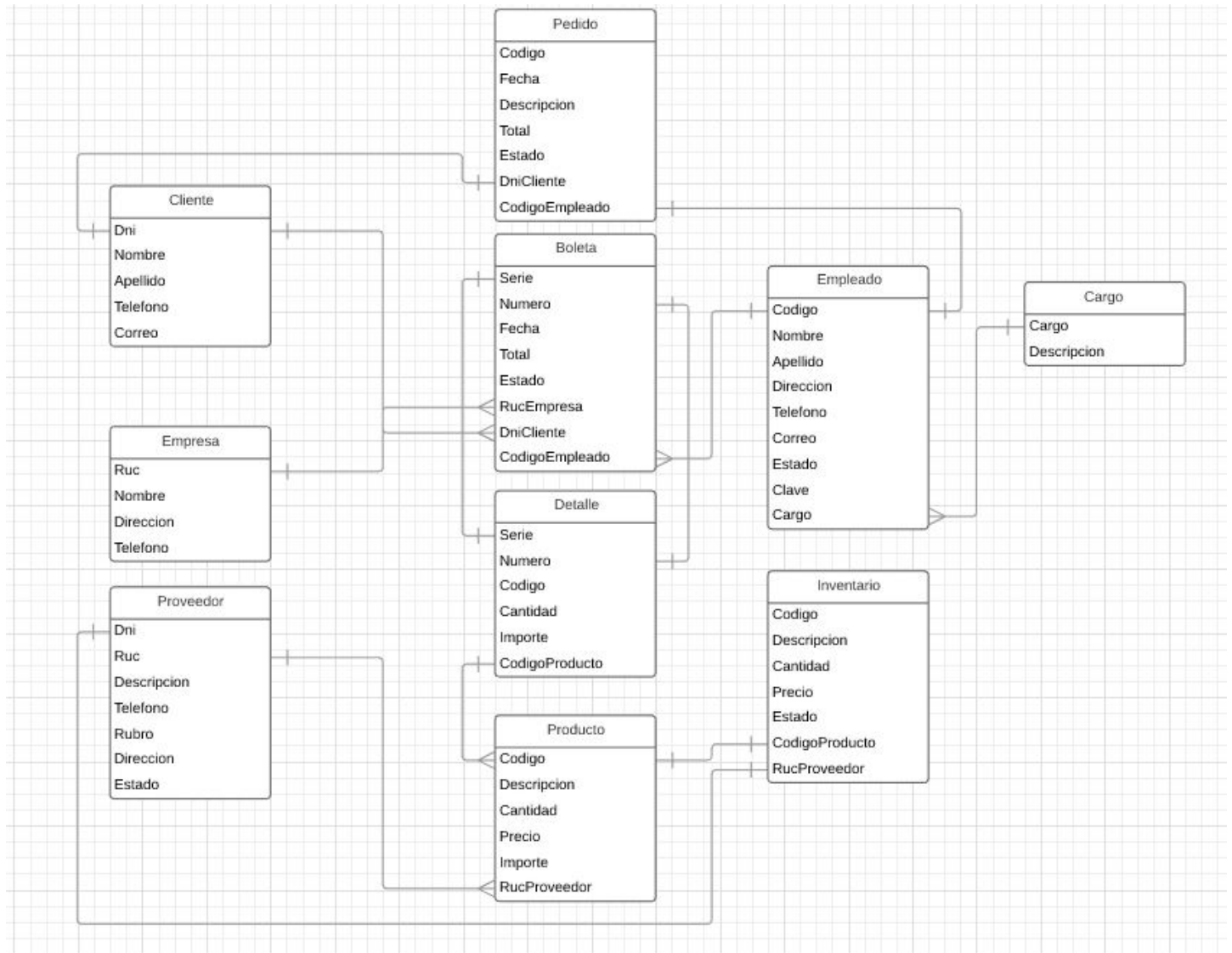
### 3.2 DISEÑO (Diagrama de Clases, Modelo Entidad Relación)

#### Diagrama de Clases





## Diagrama de Entidad/Relacion



### 3.3 PRUEBAS UNITARIAS

### 3.4 PRUEBAS POSTMAN

#### Prueba Api 1:

#### POST api/Clientes

##### Request Information

###### URI Parameters

None.

###### Body Parameters

[Clientes](#)

Name	Description	Type	Additional information
Dni		string	None.
Nombre		string	None.
Apellidos		string	None.
Telefono		string	None.
Correo		string	None.

##### Request Formats

application/json, text/json

Sample:

```
{
  "Dni": "sample string 1",
  "Nombre": "sample string 2",
  "Apellidos": "sample string 3",
  "Telefono": "sample string 4",
  "Correo": "sample string 5"
}
```

## Post:

The screenshot shows a REST client interface with the following details:

- Request Method:** POST
- URL:** http://localhost:59188/api/Clientes
- Environment:** No Environment
- Body Type:** JSON (application/json)
- Body Content:**

```
1 {  
2   "Dni": "79365487",  
3   "Nombre": "Jorge",  
4   "Apellidos": "Mamani",  
5   "Telefono": "954547854",  
6   "Correo": "racer_d3@hotmail.com"  
7 }
```
- Status:** 201 Created
- Time:** 23005 ms
- Size:** 569 B
- Body Format:** Pretty
- Body Content (Pretty):**

```
1 {  
2   "Dni": "79365487",  
3   "Nombre": "Jorge",  
4   "Apellidos": "Mamani",  
5   "Telefono": "954547854",  
6   "Correo": "racer_d3@hotmail.com"  
7 }
```

## Get:

http://localhost:59188/api/Clientes

GET http://localhost:59188/api/Clientes Send Save

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code Comments

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Body Cookies Headers (10) Test Results Status: 200 OK Time: 4852 ms Size: 510 B Down

Pretty Raw Preview JSON

```
1 [
2   {
3     "Dni": "79365487",
4     "Nombre": "Jorge",
5     "Apellidos": "Mamani",
6     "Telefono": "954547854",
7     "Correo": "racer_d3@hotmail.com"
8   }
9 ]
```

Prueba Api 2:

# POST api/Proveedores

## Request Information

### URI Parameters

None.

### Body Parameters

#### Proveedor

Name	Description	Type	Additional information
Dni		string	None.
Ruc		string	None.
Descripcion		string	None.
Telefono		string	None.
Rubro		string	None.
Direccion		string	None.
Estado		string	None.

## Request Formats

application/json, text/json

Sample:

```
{
  "Dni": "sample string 1",
  "Ruc": "sample string 2",
  "Descripcion": "sample string 3",
  "Telefono": "sample string 4",
  "Rubro": "sample string 5",
  "Direccion": "sample string 6",
  "Estado": "sample string 7"
}
```

**Post:**

http://localhost:59188/api/Proveedores

POST http://localhost:59188/api/Proveedores Send

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "Dni": "71921546",
3   "Ruc": "21387984598",
4   "Descripcion": "Arroz Costeño",
5   "Telefono": "052648949",
6   "Rubro": "Comestibles",
7   "Direccion": "Urb Tacna 65",
8   "Estado": "Activo"
9 }
```

Body Cookies Headers (11) Test Results Status: 201 Created Time: 3279 ms Size: 623 B

Pretty Raw Preview JSON

```
1 {
2   "Dni": "71921546",
3   "Ruc": "21387984598",
4   "Descripcion": "Arroz Costeño",
5   "Telefono": "052648949",
6   "Rubro": "Comestibles",
7   "Direccion": "Urb Tacna 65",
8   "Estado": "Activo"
9 }
```

Get:

http://localhost:59188/api/Proveedores

GET http://localhost:59188/api/Proveedores Send

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code

**TYPE**

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent authorization helper.

Body Cookies Headers (10) Test Results Status: 200 OK Time: 8 ms Size: 562 B

Pretty Raw Preview JSON

```
1 [
2   {
3     "Dni": "71921546",
4     "Ruc": "21387984598",
5     "Descripcion": "Arroz Costeño",
6     "Telefono": "052648949",
7     "Rubro": "Comestibles",
8     "Direccion": "Urb Tacna 65",
9     "Estado": "Activo"
10  }
11 ]
```

### Prueba Api 3:

# POST api/Productos

## Request Information

### URI Parameters

None.

### Body Parameters

#### Productos

Name	Description	Type	Additional information
Codigo		string	None.
Descripcion		string	None.
Cantidad		integer	None.
Precio		decimal number	None.
Importe		decimal number	None.
Proveedor		Proveedor	None.

## Request Formats

application/json, text/json

Sample:

```
{
  "Codigo": "sample string 1",
  "Descripcion": "sample string 2",
  "Cantidad": 3,
  "Precio": 4.1,
  "Importe": 5.0,
  "Proveedor": {
    "Dni": "sample string 1",
    "Ruc": "sample string 2",
    "Descripcion": "sample string 3",
    "Telefono": "sample string 4",
    "Rubro": "sample string 5",
    "Direccion": "sample string 6",
    "Estado": "sample string 7"
  }
}
```

Post:



POST

http://localhost:59188/api/Productos

Send

Params

Authorization

Headers [9]

Body

Pre-request Script

Tests

Cookies Co

none

form-data

x-www-form-urlencoded

raw

binary

JSON (application/json)

```

1 {
2   "Codigo": "15",
3   "Descripcion": "arroz",
4   "Cantidad": 3,
5   "Precio": 4.1,
6   "Importe": 5.0,
7   "Proveedor": {
8     "Dni": "sample string 1",
9     "Ruc": "sample string 2",
10    "Descripcion": "sample string 3",
11    "Telefono": "sample string 4",
12    "Rubro": "sample string 5",
13    "Direccion": "sample string 6",
14    "Estado": "sample string 7"
15  }
16 }

```

Body

Cookies

Headers [10]

Test Results

Status: 200 OK

Time: 3180 ms

Size: 683 B

Pretty

Raw

Preview

JSON

```

1 [
2   {
3     "Proveedor": {
4       "Dni": "sample string 1",
5       "Ruc": "sample string 2",
6       "Descripcion": "sample string 3",
7       "Telefono": "sample string 4",
8       "Rubro": "sample string 5",
9       "Direccion": "sample string 6",
10      "Estado": "sample string 7"
11    },
12    "Codigo": "15",
13    "Descripcion": "arroz",
14    "Cantidad": 3,
15    "Precio": 4.1,
16    "Importe": 5
17  }
18 ]

```

Get:

GET
http://localhost:59188/api/Productos
Send

Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body
Cookies
Headers (10)
Test Results
Status: 200 OK
Time: 3180 ms
Size: 683 B

Pretty
Raw
Preview
JSON

```

1 [
2   {
3     "Proveedor": {
4       "Dni": "sample string 1",
5       "Ruc": "sample string 2",
6       "Descripcion": "sample string 3",
7       "Telefono": "sample string 4",
8       "Rubro": "sample string 5",
9       "Direccion": "sample string 6",
10      "Estado": "sample string 7"
11    },
12    "Codigo": "15",
13    "Descripcion": "arroz",
14    "Cantidad": 3,
15    "Precio": 4.1,
16    "Importe": 5
17  }
18 ]

```

## Prueba Api 4:

# POST api/Empleados

## Request Information

### URI Parameters

None.

### Body Parameters

#### Empleados

Name	Description	Type	Additional information
Codigo		string	None.
Nombre		string	None.
Apellido		string	None.
Direccion		string	None.
Telefono		string	None.
Correo		string	None.
Estado		string	None.
Clave		string	None.
Cargo		string	None.
Cargos		Cargos	None.

## Request Formats

application/json, text/json

Sample:

```
{
  "Codigo": "sample string 1",
  "Nombre": "sample string 2",
  "Apellido": "sample string 3",
  "Direccion": "sample string 4",
  "Telefono": "sample string 5",
  "Correo": "sample string 6",
  "Estado": "sample string 7",
  "Clave": "sample string 8",
  "Cargo": "sample string 9",
  "Cargos": {
    "Cargo": "sample string 1",
    "Descripcion": "sample string 2"
  }
}
```

Post:

POST

http://localhost:59188/api/Empleados

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Cookies

Code

none

form-data

x-www-form-urlencoded

raw

binary

JSON (application/json)

```

1 {
2   "Codigo": "2016",
3   "Nombre": "Marko",
4   "Apellido": "Rivas",
5   "Direccion": "la merced 605",
6   "Telefono": "948122654",
7   "Correo": "marko@hotmail.com",
8   "Estado": "Activo",
9   "Clave": "123456",
10  "Cargo": "Vendedor",
11  "Cargos": {
12    "Cargo": "Vendedor",
13    "Descripcion": "Caja"
14  }
15 }

```

Body

Cookies

Headers (11)

Test Results

Status: 201 Created

Time: 275 ms

Size: 693 B

Pretty

Raw

Preview

JSON

```

1 {
2   "Codigo": "2016",
3   "Nombre": "Marko",
4   "Apellido": "Rivas",
5   "Direccion": "la merced 605",
6   "Telefono": "948122654",
7   "Correo": "marko@hotmail.com",
8   "Estado": "Activo",
9   "Clave": "123456",
10  "Cargo": "Vendedor",
11  "Cargos": {
12    "Cargo": "Vendedor",
13    "Descripcion": "Caja"
14  }
15 }

```

Get:

http://localhost:59188/api/Empleados

GET

http://localhost:59188/api/Empleados

Send

Save

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Cookies

Code

Comments

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Time: 30 ms

Size: 637 B

Done

Pretty

Raw

Preview

JSON



```
1 [
2   {
3     "Cargos": {
4       "Cargo": "Vendedor",
5       "Descripcion": "Caja"
6     },
7     "Codigo": "2016",
8     "Nombre": "Marko",
9     "Apellido": "Rivas",
10    "Direccion": "la merced 605",
11    "Telefono": "948122654",
12    "Correo": "marko@hotmail.com",
13    "Estado": "Activo",
14    "Clave": "123456",
15    "Cargo": "Vendedor"
16  }
17 ]
```

Prueba Api 5:

# POST api/Empresas

## Request Information

### URI Parameters

None.

### Body Parameters

#### Empresa

Name	Description	Type	Additional information
Ruc		string	None.
Nombre		string	None.
Direccion		string	None.
Telefono		string	None.

## Request Formats

application/json, text/json

Sample:

```
{
  "Ruc": "sample string 1",
  "Nombre": "sample string 2",
  "Direccion": "sample string 3",
  "Telefono": "sample string 4"
}
```

Post:

http://localhost:59188/api/Empresas

POST http://localhost:59188/api/Empresas Send

Params Authorization Headers (9) Body Pre-request Script Tests Cookies Code

none form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {
2   "Ruc": "216165489",
3   "Nombre": "Consortio A&V Distribuciones",
4   "Direccion": "Alfonso Ugarte 45 ",
5   "Telefono": "984984123"
6 }
```

Body Cookies Headers (11) Test Results Status: 201 Created Time: 2704 ms Size: 574 B

Pretty Raw Preview JSON

```
1 {
2   "Ruc": "216165489",
3   "Nombre": "Consortio A&V Distribuciones",
4   "Direccion": "Alfonso Ugarte 45 ",
5   "Telefono": "984984123"
6 }
```

Get:

GET

http://localhost:59188/api/Empresas

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Cookies

Code

TYPE

Inherit auth from parent

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

This request is not inheriting any authorization helper at the moment. Save it collection to use the parent's authorization helper.

Body

Cookies

Headers (10)

Test Results

Status: 200 OK

Time: 32 ms

Size: 514 B

Pretty

Raw

Preview

JSON

1

[

2

{

3

"Ruc": "216165489",

4

"Nombre": "Consortio A&V Distribuciones",

5

"Direccion": "Alfonso Ugarte 45 ",

6

"Telefono": "984984123"

7

}

8

]