1. Lease API Overview

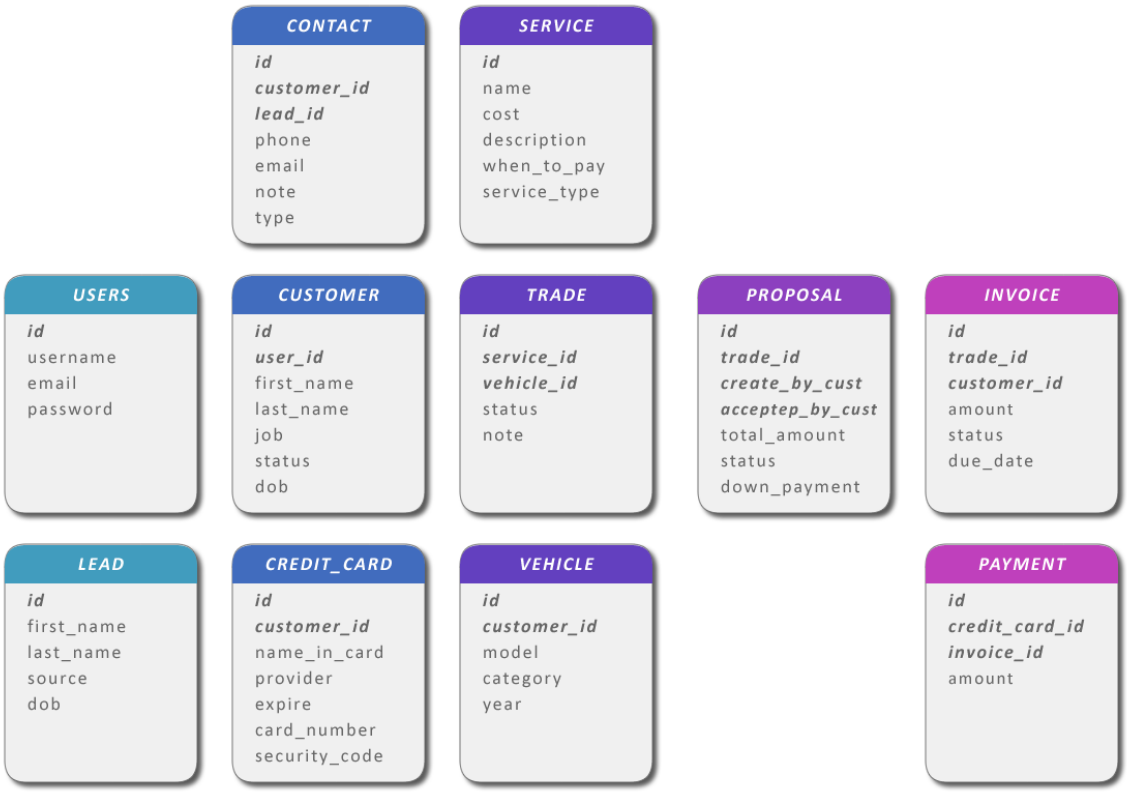
The Lease API is built for a lease company that focus on selling or leasing vehicles. The company doesn't own any vehicle but provides a platform for customers to offer their own vehicles to sell or lease and other clients to shop around.

The API is designer with Django REST API version 4.0.6, uses token authentication and is fully tested with Pytest 7.1.2 having over 300 tests for the models and the API.

Model General Design

The model consist of 11 tables. Each User can log in into the system and he/she will have a related Customer if the user is non administrative. Each Customer can store Credit Card and Contact information. Contacts can be also created for leads which can only be accessed by administrative users.

A Customer can also create a Vehicle for which he can latter create a Trade using one of the available Services (lease or sell, although you may create other services). The Customer owning the Vehicle can latter make a Proposal for that Trade and other Customer may accept that Proposal or create a new one.



If the Trade is for a Service type sell, an Invoice is created after a proposal has been accepted with the amount defined in the Service. If the Trade is created for a Service type lease (which is expected to be the most common), an Invoice is created when the first proposal is submitted. This occurs before the acceptance because the vehicle's owner is being charge for the exposure.

Finally, the customers are supposed make a Payment with a Credit Card before the due_date indicated in the Invoice. This completes the cycle around all the tables.

2. User

The User table uses the default User Django model. First you need to create a user so you can authenticate and access most of the API services.

2.1. Register User

POST https://www.domain.com/api/register/

Example request:

```
{
  "password": "Secret123*",
  "password2": "Secret123*",
  "first_name": "TestFirstName",
  "last name": "TestLastName",
  "username": "testusername",
  "email": "testemail@gmail.com",
  "job": "TestJob",
  "notes": "TestNote",
  "dob": 1970-03-04,
}
```

Example response:

```
{
  "response": "successfully registered a new user.",
  "user_id": 1916,
  "email": "TestEmail@gmail.com",
  "username": "testusername2",
  "customer id": 1811,
  "token": "5e01356d9fcd4de865b6be357a6c401b24a4a82d"
}
```

Please notice that a token is returned upon successful user registration. This token will be necessary for authentication of future requests.

2.2. Login User

POST https://www.domain.com/api/login/

A user that's already registered needs to provide his username and password to get a new token generated.

Example request:

```
{
  "username": "testusername",
  "password": "Secret123*",
}
```

Example response:

```
{
  "response": "successful login"
  "username": "testusername",
  "token": "5e01356d9fcd4de865b6be357a6c401b24a4a82d",
}
```

2.3. Get User Details

GET https://www.domain.com/api/user/{user_id} Authorization: Token 5e01...

To update get user details, the user needs to provide the Authorization Header with his Token. Each regular user can only get his own details. Staff and superusers can get any user details

Example request:

```
{

}
```

Example response:

```
{ "id": 1916
  "username": "testusername",
  "email": "TestEmail@gmail.com",
  "first_name": "TestFirstName",
  "last_name": "TestLastName",
  "is_active": true,
  "customer_id": 1811}
```

2.4. Update User

PUT https://www.domain.com/api/user/{user_id} Authorization: Token 5e01...

To update the user details, the user needs to provide the Authorization Header with his Token. Each regular user can only update his own details. Staff and superusers can get any user details.

Example request:

```
{
  "first_name": "Garret",
  "last_name": "Osborn",
  "username": "garret499",
  "email": "garret499@gmail.com",
  "is_active": 1,
}
```

Example response:

```
{ "id": 1811,
  "username": "garret499",
  "email": "garret499@gmail.com",
  "first_name": "Garret",
  "last_name": "Osborn",
  "is_active": true,
  "customer_id": 1811}
```

2.5. Update User Password

PUT https://www.domain.com/api/password_update/{user_id} Authorization: Token 5e01...

A regular user can update his own password. Staff and superusers can update any password.

Example request:

```
{
  "password ": "NewPassword123*"
}
```

Example response:

```
{
  "response": "Password Updated",
}
```

2.6. Get User List

GET https://www.domain.com/api/users/ Authorization: Token 5e01...

Only staff and superusers can get the list of all users.

Example request:

```
{

}
```

Example response:


```
[
  {
    "id": 1806,
    "username": "april599",
    "email": "april599@gmail.com",
    "first name": "April",
    "last name": "Smith",
    "is_active": true,
    "customer id": 1796
  },
  {
    "id": 1807,
    "username": "brandon686",
    "email": "brandon686@gmail.com",
    "first name": "Brandon",
    "last name": "Bell",
    "is_active": false,
    "customer id": 1804
  }, …
]
```


2.7. Delete User

DELETE https://www.domain.com/api/user/{user_id} Authorization: Token 5e01...

A regular user can delete his own user. Staff and superusers can delete any password. Deleting a user will also delete the related customer if any.

Example request:

```
{

}
```

Example response:


```
  {
    "response": "Remove Completed",
  }
```

2.8. Add User

POST https://www.domain.com/api/user/ Authorization: Token 5e01...

Only the superuser is allowed to create users using this method. It's suitable for adding users of any kind: regular, staff or superuser. Regular users are supposed to create their users by registering.

Example request:

```
{
  "password": "Secret123*",
  "is_superuser": false,
  "first_name": "Lisa",
  "last_name": "Thompson",
  "username": "lisa950",
  "email": "lisa950@gmail.com",
  "is_active": 1,
}
```

Example response:

```
{
  "id": 1917,
  "last_login": null,
  "is_superuser": false,
  "username": "lisa950",
  "first_name": "Lisa",
  "last_name": "Thompson",
  "email": "lisa950@gmail.com",
  "is_staff": false,
  "is_active": true,
  "date_joined": "2022-08-31T16:27:13.527005-04:00",
  "groups": [],
  "user_permissions": []
}
```

Please notice that when registering the request must specify the fields job, dob and notes which are not specified when adding. That's because those three fields are actually stored in the Customer table. When you create via register, a Customer is also created. On the contrary, creating a user via add only creates a record in the User table.

2.9. User Search

GET https://www.domain.com/api/users/search/?username=lis Authorization: Token 5e01...

Only the staff and superuser are allowed to search users. Search can be made by username, first_name and/or last_name. Note that the response indicates customer_id = 0 if there is the user has no related customer

Example request:

```
{

}
```

Example response:

```
[{
    "id": 1917,
    "username": "lisa950",
    "email": "lisa950@gmail.com",
    "first_name": "Lisa",
    "last_name": "Thompson",
    "is_active": true,
    "customer id": 0
}]
```