

Décisions prises

Modèle de données

Service d'authentification – Table Users

- Contient les champs requis et des champs optionnels selon mes besoins
- Les mots de passe sont stockés de façon sécurisée grâce à **bcrypt**.
- Le champ **revenu** est aussi enregistré (par exemple, lors d'une demande de prêt).

Service de prêts – Table Loans

- Un champ **remaining_amount** est utilisé pour suivre en temps réel le montant restant à rembourser.
- Les prêts peuvent avoir différents **statuts** (ex. : en attente, approuvé, remboursé) avec des transitions entre eux

Service de paiements – Table Payments

- cest pour prendre en charge plusieurs méthodes de paiement (ex. : carte, virement, portefeuille numérique).
- Chaque paiement est identifié par un **ID de transaction** généré et stocké dans la database.
- Chaque enregistrement de paiement est lié à un prêt spécifique et à l'utilisateur correspondant.

Explications

Règles de validation mises en place

Pour éviter les abus ou les risques trop élevés, j'ai mis en place quelques règles de base côté backend :

- Montant maximal de 5000\$
- Revenu mensuel supérieur à 1200\$

- Date d'échéance dans le futur : logique, on ne peut pas avoir une date de fin dans le passé.
- Maximum de 3 prêts actifs par utilisateur juste pour éviter trop d'endettement

Règles supplémentaires

J'ai aussi ajouté quelques validations en plus du genre:

- Montant minimal de 100\$
- Durée maximale de 2 ans
- Vérification de l'historique de crédit si le prêt dépasse 1000\$
- Total des prêts inférieur ou égal 3 x le revenu mensuel

5. Fonctionnement des remboursements

Remboursements partiels

J'ai décidé de permettre les paiements partiels, car ça donne plus de flexibilité à l'utilisateur. À chaque paiement, le champ `remaining_amount` est automatiquement mis à jour, donc le système garde toujours une trace précise de ce qu'il reste à payer.

Remboursements anticipés

Ils sont aussi possibles. En mode l'idée, c'est d'encourager les gens à rembourser dès qu'ils peuvent.

Mécanisme de transaction

Pour les paiements, j'ai mis en place une logique des transactions: si une étape échoue (genre la mise à jour du prêt ou l'enregistrement du paiement), tout est annulé avec un `rollback`.

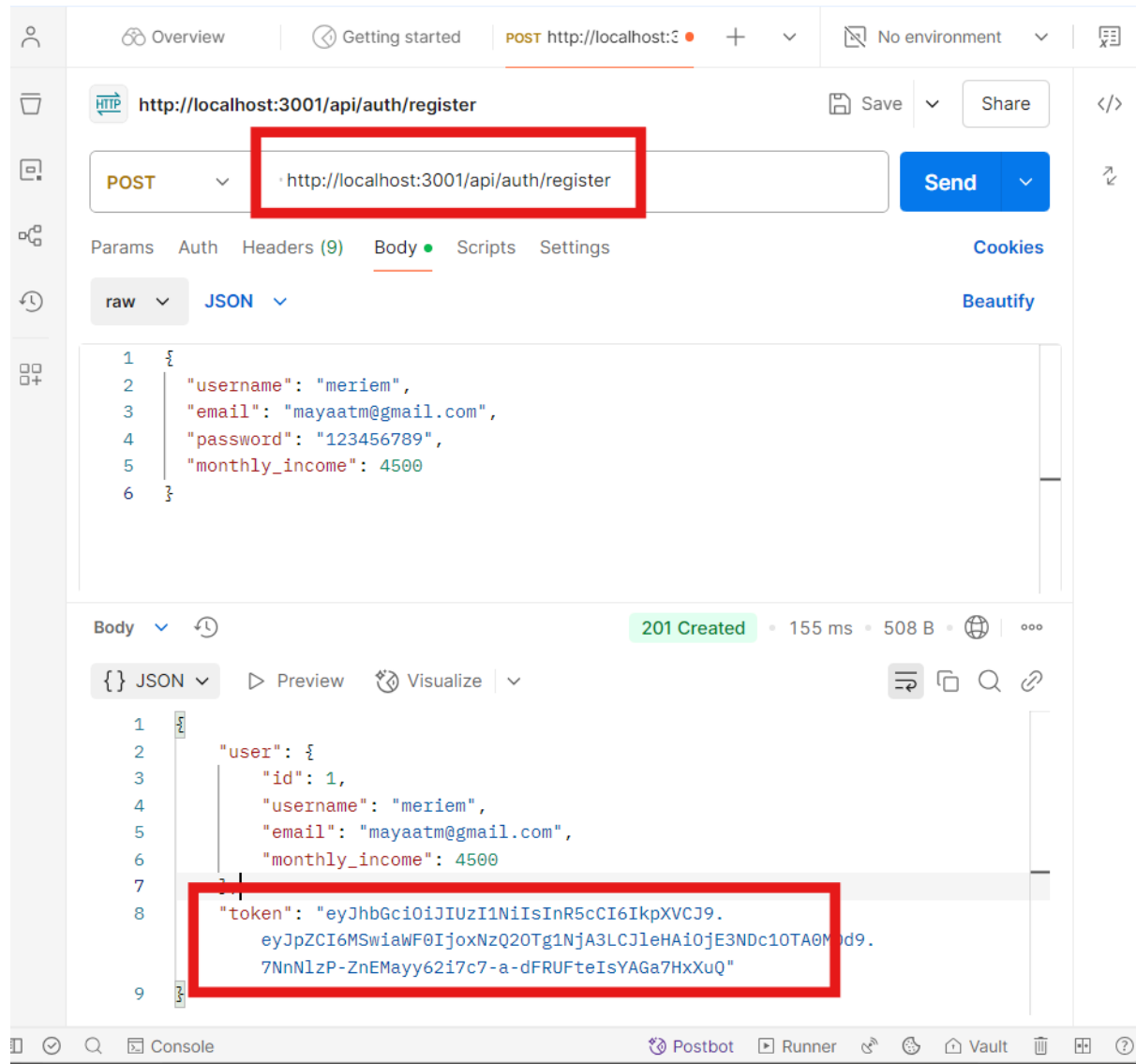
6. Sécurité

Authentification (JWT)

Chaque token a une date d'expiration, et seul l'utilisateur connecté peut accéder à ses propres données. Ça permet une bonne sécurisation des accès sans trop complexifier l'architecture.

Capture d'écrans

1. Inscription d'un utilisateur



2. Connexion d'un utilisateur

HTTP <http://localhost:3001/api/auth/login> Save Share

POST <http://localhost:3001/api/auth/login> Send

Params Auth Headers (9) Body Scripts Settings Cookies

raw JSON Beautify

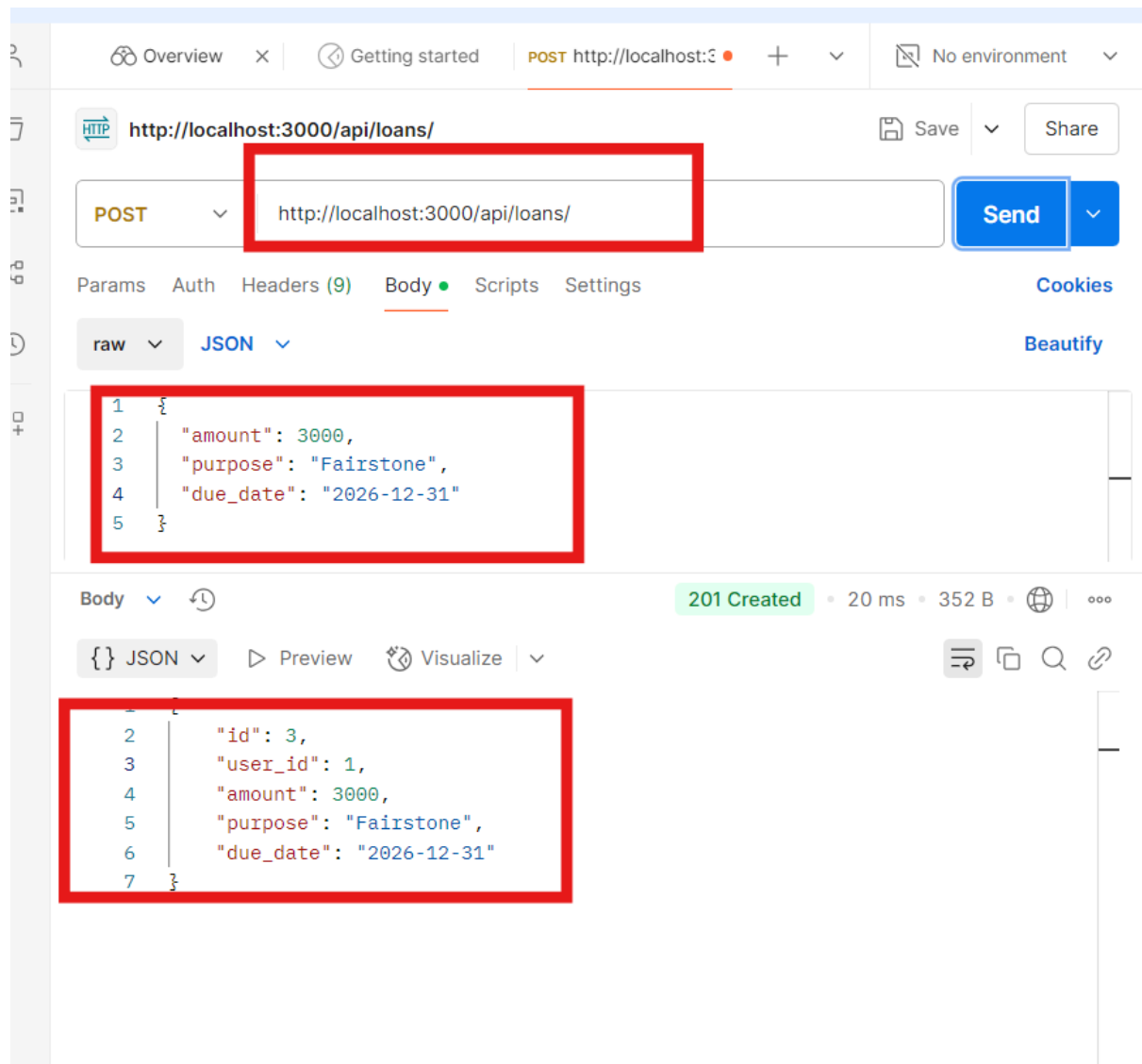
```
{
  "username": "meriem",
  "password": "123456789"
}
```

Body 200 OK • 107 ms • 503 B • 🌐 ⋮

JSON Preview Visualize

```
1 {
2   "user": {
3     "id": 1,
4     "username": "meriem",
5     "email": "mayaatm@gmail.com",
6     "monthly_income": 4500
7   },
8   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNzQ2OTg1NzIxLCJleHAiOjE3NDc1OTA1MjF9.Lf-YTEfQAp-2hM0701fq_u1_SsWSHpUwbBtp8sbfrBU"
9 }
```

3.Soumission d'un prêt (Réussite)



4. Échec de soumission d'un prêt (Montant dépassé)

HTTP <http://localhost:3000/api/loans/> Save Share

POST <http://localhost:3000/api/loans/> Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

raw JSON Beautify

```
1 {
2   "amount": 13000,
3   "purpose": "Fairstone",
4   "due_date": "2026-12-31"
5 }
```

Body 400 Bad Request • 9 ms • 319 B • 🌐 ⋮

{ } JSON Preview Visualize

```
1 {
2   "error": "Loan amount cannot exceed $5000"
3 }
```

Échec de soumission d'un prêt (Validation de date)

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

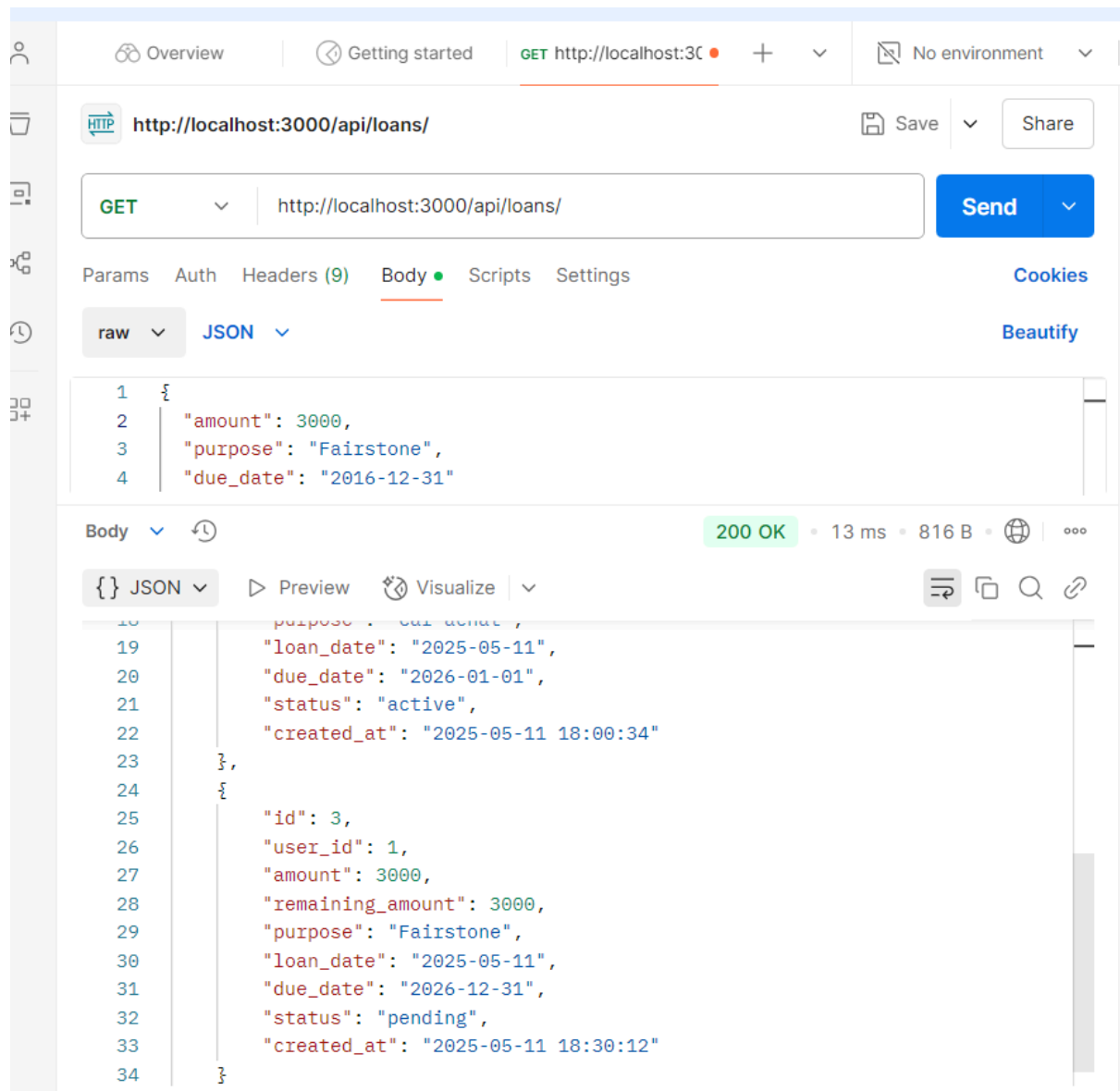
- URL:** `http://localhost:3000/api/loans/`
- Method:** `POST`
- Body (JSON):**

```
1 {  
2   "amount": 3000,  
3   "purpose": "Fairstone",  
4   "due_date": "2016-12-31"  
5 }
```
- Response:** `400 Bad Request` (7 ms, 316 B)
- Response Body (JSON):**

```
1 {  
2   "error": "Due date must be after today"  
3 }
```

The response body is highlighted with a red rectangle.

6. Consultation des prêts (juste un GET)



7. Effectuer un paiement réussi (paiement partiel)

HTTP <http://localhost:3000/api/payments> Save Share

POST <http://localhost:3000/api/payments> Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

raw JSON Beautify

```
1 {
2   "loan_id": 2,
3   "amount": 2500,
4   "payment_method": "credit_card"
5 }
```

Body 201 Created • 39 ms • 386 B • Preview Visualize

```
1 {
2   "id": 3,
3   "loan_id": 2,
4   "user_id": 1,
5   "amount": 2500,
6   "payment_method": "credit_card",
7   "transaction_id": "dedf10813317f4d2"
8 }
```

on verifier que 2500 ont été déduits

The screenshot shows a REST client interface. At the top, the URL bar displays `http://localhost:3000/api/loans/3`. Below it, the method is set to `GET`. The request body is empty. The response is shown in the 'Body' tab, displaying a JSON object: `{ "loan_id": 3, "amount": 2500, "payment_method": "credit_card" }`. Below this, a second JSON object is shown, representing the full loan record: `{ "id": 3, "user_id": 1, "amount": 3000, "remaining_amount": 500, "purpose": "Fairstone", "loan_date": "2025-05-11", "due_date": "2026-12-31", "status": "active", "created_at": "2025-05-11 18:30:12" }`. The status bar at the bottom indicates a `200 OK` response with a response time of 8 ms and a body size of 449 B.

```
GET http://localhost:3000/api/loans/3
```

```
{
  "loan_id": 3,
  "amount": 2500,
  "payment_method": "credit_card"
}
```

```
{
  "id": 3,
  "user_id": 1,
  "amount": 3000,
  "remaining_amount": 500,
  "purpose": "Fairstone",
  "loan_date": "2025-05-11",
  "due_date": "2026-12-31",
  "status": "active",
  "created_at": "2025-05-11 18:30:12"
}
```

200 OK • 8 ms • 449 B

payer tous le montant (celui qui reste) 500\$

HTTP <http://localhost:3000/api/payments> Save Share

POST <http://localhost:3000/api/payments> Send

Params Auth Headers (9) Body • Scripts Settings Cookies

raw JSON Beautify

```
1 {
2   "loan_id": 3,
3   "amount": 500,
4   "payment_method": "credit_card"
5 }
```

Body Body 201 Created • 34 ms • 385 B •

{ } JSON Preview Visualize

```
1 {
2   "id": 5,
3   "loan_id": 3,
4   "user_id": 1,
5   "amount": 500,
6   "payment_method": "credit_card",
7   "transaction_id": "dfc48be1f885992f"
8 }
```

On va voir qu'il reste rien

HTTP <http://localhost:3000/api/loans/3> Save Share

GET <http://localhost:3000/api/loans/3> Send

Params Auth Headers (9) **Body** Scripts Settings Cookies Beautify

raw JSON

```
1 {
2   "loan_id": 3,
3   "amount": 500,
4   "payment_method": "credit_card"
5 }
```

Body 200 OK • 8 ms • 445 B

{ } JSON Preview Visualize

```
1 {
2   "id": 3,
3   "user_id": 1,
4   "amount": 3000,
5   "remaining_amount": 0,
6   "purpose": "Fairstone",
7   "loan_date": "2025-05-11",
8   "due_date": "2026-12-31",
9   "status": "paid",
10  "created_at": "2025-05-11 18:30:12"
11 }
```

SWAGGER

Swagger
Supported by SMARTBEAR

API BES Loan 1.0.0 OAS 3.0
API pour le système de microservices BES Loan

Servers
/api - Passerelle API BES Loan Authorize

Authentication

- POST /auth/register Inscrire un nouvel utilisateur
- POST /auth/login Connecter un utilisateur existant

Prêts

- POST /loans Créer une nouvelle demande de prêt
- GET /loans Obtenir tous les prêts de l'utilisateur authentifié
- GET /loans/{id} Obtenir un prêt spécifique par son ID

Authentication

POST /auth/register

Inscrire un nouvel utilisateur

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "username": "john doe",
  "email": "john@example.com",
  "password": "motDePasseSecurise123",
  "monthly_income": 3500
}
```

Responses

Code	Description	Links
201	Utilisateur inscrit avec succès	No links

Media type

application/json

authentification

POST /auth/register

Inscrire un nouvel utilisateur

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "username": "maya",
  "email": "maymay@hotmail.com",
  "password": "mdp123456789",
  "monthly_income": 6500
}
```

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/auth/register' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "username": "maya",
    "email": "maymay@hotmail.com",
    "password": "mdp123456789",
    "monthly_income": 6500
  }'
```

Request URL

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/auth/register' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "username": "maya",
    "email": "maymay@hotmail.com",
    "password": "mdps123456789",
    "monthly_income": 6500
  }'
```

Request URL

http://localhost:3000/api/auth/register

Server response

Code

Details

201

Response body

```
{
  "user": {
    "id": 1,
    "username": "maya",
    "email": "maymay@hotmail.com",
    "monthly_income": 6500
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNjQ0MjQ0MD09.Rct9Dj3d88o7j1Gj1Qw0Dv4lrr-JoYFKdVwQfG3c"
}
```

Download

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 234
content-type: application/json; charset=utf-8
date: Sun, 11 May 2025 18:53:20 GMT
etag: W/"ea-r2HfFPWjte86cZuMwG5NIvss"
keep-alive: timeout=5
x-powered-by: Express
```

Responses

Code	Description	Links
201	Utilisateur inscrit avec succès	No links

LOGIN

POST /auth/login Connecter un utilisateur existant

Parameters

Cancel

Reset

No parameters

Request body required

application/json

```
{
  "username": "maya",
  "password": "mdps123456789"
}
```

Execute

```
curl -X 'POST' \
  'http://localhost:3000/api/auth/login' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "username": "mays",
    "password": "mdp123456789"
  }'
```

```
http://localhost:3000/api/auth/login
```

Code	Details
------	---------

Response body

```
{
  "user": {
    "id": 1,
    "username": "maya",
    "email": "mayamay@hotmail.com",
    "monthly_income": 6500
  },
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC9i.eyJpZCI6ImlwYWVlOTJodnZjZWZgNjc1LmElaW10JE3NDCl0TQW9kN9.pH0931zF0am3icfYtU9cmK78qX6q-nly/zp-TdMeB0"
```

```
access-control-allow-origin: *
connection: keep-alive
content-length: 234
content-type: application/json; charset=utf-8
date: Sun, 11 May 2025 18:54:33 GMT
etag: W/"ea-f040bb7c1261d7238kodH4ZY"
keep-alive: timeout=5
x-powered-by: Express
```

Code	Description	Links
------	-------------	-------

Utilisateur connecté avec succès

No links

application/json

Parameters Cancel Reset

Execute

Clear

Responses

curl

```
curl -X 'POST' \
  'http://localhost:3000/api/loans' \
  -H 'accept: application/json' \
  -H 'Authorization: bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IWRXVz9yZ2ZlbnR1bWVudW8iOiJ0d20tZS5NcmlzIiwiaWF0IjE3MDc1OTQ0Nz09.p4D931zF0m3KcFyE19c1k78qX6q-nlyZp9-TdWebU' \
  -H 'Content-Type: application/json' \
  -d '{
    "amount": 1000,
    "purpose": "renov maison",
    "due_date": "2026-01-01"
  }'
```

Request URL

http://localhost:3000/api/loans

Server response

Code	Details
201	<div><div>Response body</div><div><pre>{ "id": 4, "user_id": 2, "amount": 1000, "purpose": "renov maison", "due_date": "2026-01-01" }</pre></div><div>Response headers</div><div><pre>access-control-allow-origin: * connection: keep-alive content-length: 83 content-type: application/json; charset=utf-8 date: Sun, 11 May 2025 18:56:32 GMT etag: W/"53-ClrchOpittDse8LP0uice/f0bxCQ" keep-alive: timeout=5 x-powered-by: Express</pre></div></div>

Responses

Code	Description	Links
201	Demande de prêt créée avec succès	No links

Media type

application/json

Controls Accept header

Example Value | Schema

GET LOANS

GET /loans Obtenir tous les prêts de l'utilisateur authentifié

Cancel

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3000/api/loans' \
  -H 'accept: application/json' \
  -H 'Authorization: bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IWRXVz9yZ2ZlbnR1bWVudW8iOiJ0d20tZS5NcmlzIiwiaWF0IjE3MDc1OTQ0Nz09.p4D931zF0m3KcFyE19c1k78qX6q-nlyZp9-TdWebU'
```

Request URL

http://localhost:3000/api/loans

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "id": 4, "user_id": 2, "amount": 1000, "remaining_amount": 1000, "purpose": "renov maison", "loan_date": "2025-05-11", "due_date": "2026-01-01", "status": "pending", "created_at": "2025-05-11 18:56:32" }</pre></div><div>Response headers</div><div><pre>access-control-allow-origin: * connection: keep-alive content-length: 188 content-type: application/json; charset=utf-8 date: Sun, 11 May 2025 18:57:29 GMT etag: W/"bc-yx01Ve/0r83DSMT95wb3VxfnqJI" keep-alive: timeout=5 x-powered-by: Express</pre></div></div>

Responses

Code	Description	Links
200	Liste des prêts	No links

GET UN LOAN AVEC SON ID

on va essayer d'accéder au prêt de lit 3 (on est authentifié en tant que id 4)
techniquement on a pas le droit

Name

Description

id * required
integer
(path)

ID du prêt

3

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  "http://localhost:3000/api/loans/3" \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc2UiOiJlbnVlcnF0IiwiaWF0IjoxNjQyOTg5Mjc1LmEHA1OjE3NDc1OTQwZnR9.pdD931zf0am3hcftYEU9ckk78qX6q-nUyZp9-TxdMbU"
```

Request URL

```
http://localhost:3000/api/loans/3
```

Server response

Code

Details

403

Error: Forbidden

Response body

```
{
  "error": "Not authorized to access this loan"
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 46
content-type: application/json; charset=utf-8
```

message not authorized affiché

Effectuer un payment

PAYMENT (BAREAR id4 sans le body) et on paie 200

Paielements

POST /payments Effectuer un paiement pour un prêt

Parameters

No parameters

Request body required

application/json

```
{  "loan_id": 4,  "amount": 200,  "payment_method": "credit_card"}
```

Execute

Clear

Curl

```
curl -X 'POST' \
  "http://localhost:3000/api/payments" \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZiI6ImVudW81IiwiaWF0IjoxNzQ2OTg5NjczLjE4MDEwE3B0c1OTQ0NzN9.pH931zF0am3icfyfEU9cck78qK6q-nbyZp9-Tx9ebu' \
  -H 'Content-Type: application/json' \
  -d '{
    "loan_id": 4,
    "amount": 200,
    "payment_method": "credit_card"
  }'
```

Request URL

http://localhost:3000/api/payments

Server response

Code	Details
201	<p>Response body</p> <pre>{ "id": 5, "loan_id": 4, "user_id": 2, "amount": 200, "payment_method": "credit_card", "transaction_id": "3b413dbcabe00430" }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 112 content-type: application/json; charset=utf-8 date: Sun, 11 May 2025 19:06:17 GMT etag: W/"70-AZmzZooTzmKwR0dEvSr8yda" keep-alive: timeout=5 x-powered-by: Express</pre>

Responses

Code	Description	Links
201	Paiement traité avec succès	No links

GET UN LOAN

GET /payments/loan/{loanId} Obtenir les paiements pour un prêt spécifique

Parameters

Cancel

Name	Description
loanId * required	ID du prêt
integer (path)	4

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  "http://localhost:3000/api/payments/loan/4" \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZiI6ImVudW81IiwiaWF0IjoxNzQ2OTg5NjczLjE4MDEwE3B0c1OTQ0NzN9.pH931zF0am3icfyfEU9cck78qK6q-nbyZp9-Tx9ebu'
```

Request URL

http://localhost:3000/api/payments/loan/4

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 5, </pre>

si on vérifie les détails du pret on vois qu'il reste 800 a payer mtn

Curl

```
curl -X 'GET' \
  'http://localhost:3000/api/loans/4' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImwiYW90IjoxNzQ2OTg5Njc3LCJleHAiOiJlE3NDc1OTQ0NzN9.pH'
```

Request URL

http://localhost:3000/api/loans/4

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": 4, "user_id": 2, "amount": 1000, "remaining_amount": 800, "purpose": "renov maison", "loan_date": "2025-05-11", "due_date": "2026-01-01", "status": "active", "created_at": "2025-05-11 18:56:32" }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 184 content-type: application/json; charset=utf-8 date: Sun, 11 May 2025 19:16:02 GMT etag: W/"b8-Nqzsg0lAbcZAc7TxrokLXdxKU4I" keep-alive: timeout=5 x-powered-by: Express</pre>

Responses

Code	Description
200	Détails du prêt

Media type

application/json

le shema

Schemas

- User >
- AuthResponse >
- Loan >
- Payment >
- Error >

Si j'essaye de payer un montant supérieur 5000 (la limite maximum selon moi)

ca met un internal server error

The screenshot shows a REST client interface with a JSON body for a POST request:

```
{  "loan_id": 4,  "amount": 60000,  "payment_method": "credit_card"}
```

Below the body is an "Execute" button. Under the "Responses" tab, the "Curl" section shows the equivalent curl command:

```
curl -X 'POST' \  'http://localhost:3000/api/payments' \  -H 'accept: application/json' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImwiYWwF0IjoxNzQ2OTg5Njc5LCJleHAiOjE3NDc1OTQ0NzN9.phD93lzf' \  -d '{  "loan_id": 4,  "amount": 60000,  "payment_method": "credit_card"  }'
```

The "Request URL" is `http://localhost:3000/api/payments`. The "Server response" section shows a 500 status code with the message "Error: Internal Server Error".

Paieement SI LE TOKEN EST ABSENT
jle laisse vide sans token

The dialog box titled "Available authorizations" shows the "BearerAuth (http, Bearer)" type. The "Value:" field is empty. At the bottom are "Authorize" and "Close" buttons.

jessaye de faire le payment

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/payments' \
  -H 'accept: application/json' \
  -H 'content-type: application/json' \
  -d '{
    "loan_id": 4,
    "amount": 200,
    "payment_method": "credit_card"
  }'
```

Request URL

```
http://localhost:3000/api/payments
```

Server response

Code

Details

401

Error: Unauthorized

Response body

```
{
  "error": "Authentication required"
}
```

Download

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 35
content-type: application/json; charset=utf-8
date: Sun, 11 May 2025 19:21:15 GMT
etag: W/"23-SuCR8hagvtvEDNlkkfKw0BN"
keep-alive: timeout=5
x-powered-by: Express
```

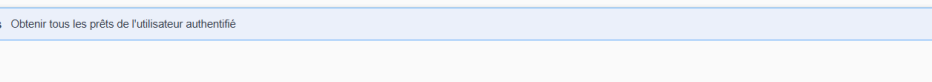
Responses

Code

Description

Links

Aussi toute mes routes sont securisé avec les barear token est demande lauthentification



The screenshot displays the Swagger UI for the `/loans` endpoint. The interface includes a header with the endpoint path and the HTTP method (GET). The description of the endpoint is 'Obtenir tous les prêts de l'utilisateur authentifié'. The 'Parameters' section is empty. The 'Responses' section shows a 200 status code with a JSON response. The 'Execute' button is highlighted.

Available authorizations

x

BearerAuth (http, Bearer)

Authorized

Value: *****

Logout

Close