# RANCANG BANGUN *REST API* APLIKASI PENCATATAN KEUANGAN BERBASIS WEB MENGGUNAKAN PENERAPAN MONGOOSE

**Melati Nour Esa, Apriade Voutama**
Sistem Informasi, Universitas Singaperbangsa Karawang
Jl. HS. Ronggo Waluyo, Puseurjaya, Kec. Telukjambe Timur, Karawang, Jawa Barat 41361, Indonesia.
*melatijbg@gmail.com*

**ABSTRAK**

Mengatur manajemen keuangan pribadi adalah kemampuan yang perlu dilatih sejak dini agar individu dapat mengolah pemasukan dan pengeluarannya dengan seimbang. Memiliki catatan keuangan yang baik, dapat memudahkan individu untuk menentukan prioritas dalam pengaturan keuangan mereka. Masalah utama dalam manajemen keuangan pribadi adalah pencatatan keuangan masih menggunakan metode manual yang tidak praktis dan membuat individu khawatir akan lebih mudahnya kehilangan data transaksi. Tujuan membuat rancang bangun *REST API* adalah untuk meningkatkan keamanan informasi atau data dalam sistem pada aplikasi pencatatan keuangan. Penelitian ini menggunakan metode *Agile Software Development* dan *pemodelan Unified Modeling Language (UML)*. Hasil penelitian ini menunjukkan bahwa sistem berhasil mengatasi masalah kekhawatiran individu mengenai mudahnya kehilangan data atau informasi transaksi.

***Kata kunci:*** *Mongoose, REST API, Catatan keuangan*

## 1. PENDAHULUAN

Mengatur manajemen keuangan pribadi adalah kemampuan yang perlu dilatih sejak dini agar individu dapat mengolah pemasukan dan pengeluarannya dengan seimbang. Banyak cara yang dapat dilakukan individu untuk melatih kemampuannya dalam mengelola keuangan pribadi[1]. Salah satunya ialah membuat catatan untuk setiap transaksi yang terjadi. Pencatatan keuangan dapat memudahkan individu untuk mengendalikan pengeluaran serta pemasukannya. Adanya catatan keuangan yang baik, individu dapat dengan mudah menentukan prioritas dalam pengaturan keuangan mereka[2]. Namun, cara manual tersebut tidak praktis dan seringkali individu merasa khawatir karena data transaksi akan lebih mudah hilang.

Saat ini, teknologi informasi sudah berkembang secara pesat. Tentunya, hal tersebut menjadi peluang yang bagus untuk mengembangkan inovasi baru dalam mengatasi permasalahan yang terjadi[3]. Salah satu upaya untuk mengatasi masalah tersebut adalah membuat rancangan sistem manajemen keuangan berbasis *Application Programming Interface (API)* sebagai *Backend Development* yang dapat diimplementasikan pada *website* atau *mobile* berbasis android dan akan bertanggung jawab baik dari sisi *server* maupun *database*[4]. Tujuan membuat rancang bangun *REST API* adalah untuk meningkatkan keamanan informasi atau data dalam sistem pada aplikasi pencatatan keuangan. Dalam penelitian ini, akan mengembangkan *Representatif State Transition Application Programming Interface (REST API).*

*REST* merupakan arsitektur perangkat lunak (*software)* yang memiliki protokol atau aturan komunikasi berbasis standar web. Sedangkan, *API* adalah rancangan antarmuka yang dibuat oleh pengembang sehingga komponen dan fungsi sistem dapat diakses dengan program. Dengan menggunakan API yang aman dan terenkripsi, aplikasi manajemen keuangan pribadi dapat melindungi data pribadi pengguna[5]. Selain itu, API juga akan menjaga data keuangan pengguna dengan baik.

Dalam perancangannya, akan menggunakan salah satu *Database Management System (DBMS) NoSQL* yaitu MongoDB sebagai tempat menyimpan data-data pengguna dalam format dokumen JSON. Selain itu, impelementasi *Mongoose* diterapkan agar dapat berinteraksi dengan MongoDB dalam aplikasi Node.js.

## 2. TINJAUAN PUSTAKA

Penelitan serupa dilakukan oleh Fransiskus Fernando dkk, tahun 2023. Penelitian tersebut mengenai Aplikasi Manajemen Keuangan Berbasis Android Terintegrasi Sistem Pengingat dan Pencatat Keuangan. Penelitian ini melibatkan beberapa tahap dalam pembuatannya yaitu membuat rancangan penelitian menggunakan desain penelitian deskriptif dengan mempelajari literatur-literatur yang berkaitan dengan penelitian guna mengetahui cara merancang aplikasi manajemen keuangan yang menghubungkan penjadwalan dan catatan keuangan, teknik pengumpulan data dengan cara mencari sumber-sumber yang berkaitan dengan aplikasi yang dibangun dan memperkuat teori-teori yang telah tersedia agar memberikan data yang valid, teknik analisis sistem menggunakan teknik berorientasi objek. Alat yang digunakan sebagai pemodelan sistem adalah *Unified Modeling Language* (UML) yang memiliki peran untuk memvisualisasikan prosedur dan aliran data yang terdapat pada rancangan aplikasi manajemen keuangan, dan aplikasi perancangan sistem. Penelitian ini menghasilkan sebuah aplikasi manajemen keuangan yang sangat membantu pengguna dalam mengatur pengeluaran dan menabung. Aplikasi ini membantu pengguna dalam memantau kondisi

keuangan dengan jelas dan mudah dipahami dengan adanya penyajian total pengeluaran, pemasukan dan tabungan[2].

Penelitian yang dilakukan oleh Muhammad Darwis dkk 2023, mengenai Pengembangan Aplikasi MyNeeds Berorientasi Objek Untuk Memantau Keuangan Mahasiswa. Metode penelitian yang digunakan dalam penelitian ini adalah metode dan pendekatan berorientasi objek yang memiliki 5 tahapan diantaranya, identifikasi masalah dengan mengidentifikasi masalah yang dialami mahasiswa dalam mengatur keuangan mereka, pengumpulan data yang dilakukan untuk menemukan solusi terhadap permasalahan dan berasal dari hasil observasi langsung serta hasil interview mahasiswa, perancangan aplikasi MyNeeds dan memanfaatkan fitur dan *tools* UML antara lain *use case*, *class* dan *activity diagram* untuk memudahkan dalam proses perancangan, implementasi aplikasi menggunakan bahasa pemrograman *Dart* dalam *framework* Flutter yang mudah digunakan untuk merancang aplikasi berbasis android, dan pengujian aplikasi. Penelitian ini juga menggunakan pemodelan sistem *Unified Modelling Language (UML)*. Terbatasnya pendapatan dan besar kebutuhan menjadi tantangan untuk mahasiswa dalam pengelolaan keuangan mereka. Kebutuhan *monitoring* keuangan akan sangat membantu mahasiswa dalam mencatat pemasukan dan memantau pengeluaran mereka secara efektif dan efisien. Aplikasi tersebut telah melewati masa pengujian dengan menggunakan metode *blackbox* dan fungsi setiap fitur telah berjalan dengan baik sehingga dapat digunakan oleh mahasiswa dalam pengelolaan keuangan mereka[1].

Selanjutnya, Muhammad dan Irving V. Paputungan juga melakukan penelitian mengenai Pengembangan Backend Server Berbasis Arsitektur REST API pada Sistem Transfer Dompet Digital. Penelitian ini menggunakan metode *Extreme programming* (XP) yang memiliki fokus pada pengembangan kode. *Extreme programming* (XP) memiliki beberapa tahapan dalam proses pengembangan nya yaitu perencanaan, perancangan, pengkodean, dan pengujian[5].

## 3. METODE PENELITIAN

*Agile Software Development* adalah pendekatan dalam pengembangan *software* yang berfokus pada kerja kolaboratif, iteratif, serta responsif terhadap perubahan. Penelitian ini menggunakan salah satu model pengembangan perangkat lunak *Agile Software Development*, yaitu *Extreme Programming* (XP)[5].

*Extreme Programming* (XP) adalah metode pengembangan yang memiliki fokus pada kode atau *coding*. Selain itu, *Extreme Programming* bersifat iteratif dan dapat mengalami perulangan dalam fasenya[5]. *Extreme Programming* (XP) memiliki beberapa tahapan dalam proses pengembangannya, diantaranya:

a. Perencanaan (*Planning*), tahapan ini untuk memahami dan menentukan fitur-fitur yang dibutuhkan seperti fitur membuat catatan keuangan baru, mengubah catatan keuangan, dan menghapus catatan keuangan.
b. Perancangan (*Design*), tahapan ini dilakukan untuk mendefinisikan setiap komponen aplikasi catatan keuangan.
c. Koding (*Coding*), tahap yang memasuki pembuatan kode sesuai dengan fungsional fitur yang sudah ditentukan dalam tahapan *planning.*
d. Pengujian (*Testing*), tahap uji coba fungsionalitas serta respon API aplikasi catatan keuangan.

Dalam penelitian ini, rancangan REST API pada aplikasi pencatatan keuangan berbasis web terbagi menjadi tiga bagian, diantaranya rancangan *Backend Development*, rancangan arsitektur sistem, dan rancangan *API*. Berikut adalah proses awal perancangan dari *Backend Development* :

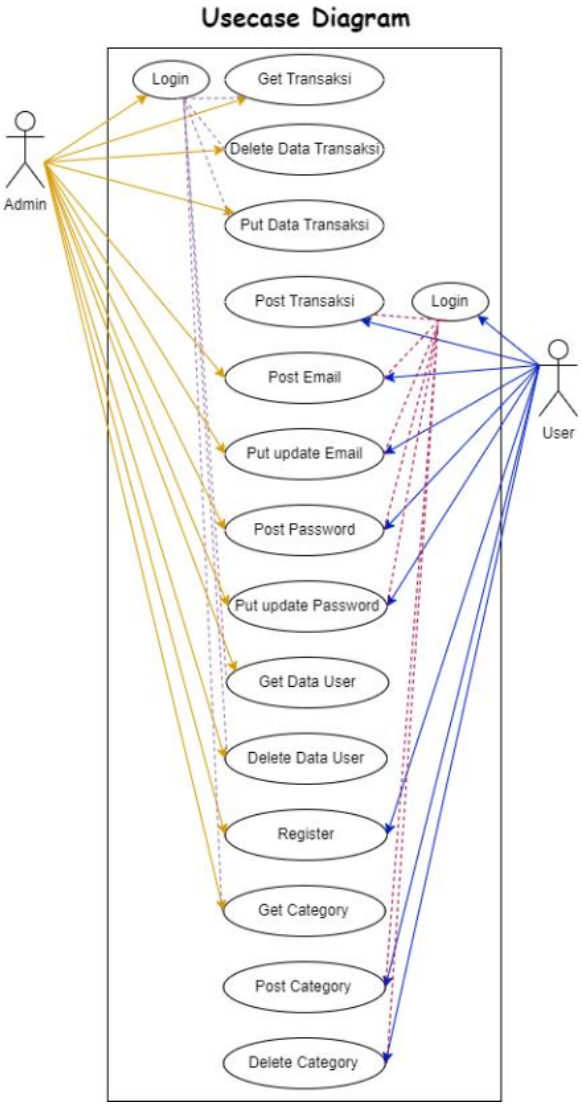
### 3.1 Rancangan *Backend Development*

*Backend Development* merupakan proses merancang sistem yang dilakukan di balik layar dari sebuah website atau aplikasi dan tidak dapat terlihat secara langsung oleh pengguna. Rancangan ini berfokus pada *database, scripting,* dan juga arsitektur dari sebuah website atau aplikasi.

Sebelum merancang, penelitian ini membuat beberapa rancangan model sistem dengan menggunakan *Unified Modelling Language* (UML) sebagai gambaran atau dokumentasi dari sistem aplikasi pencatatan keuangan. Model sistem yang digunakan terdiri dari *usecase diagram*, *activity diagram*, dan *Entity Relantionship Diagram* (ERD).

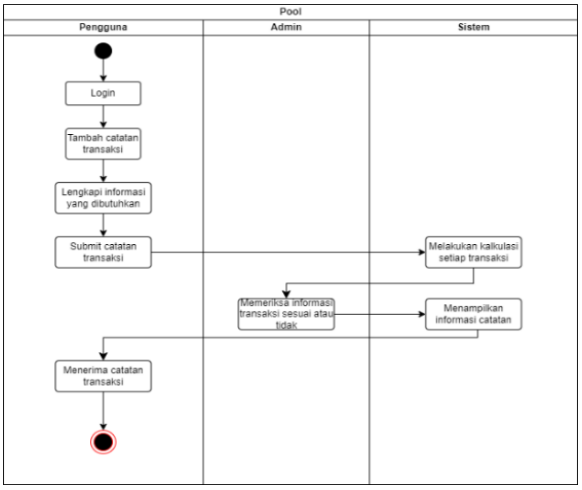
*Usecase diagram* adalah gambaran dari bentuk interaksi suatu sistem yang menghubungkan aktor dengan sistem. Rancangan *usecase diagram* dalam penelitian ini memiliki 2 aktor yaitu pengguna (*user)* dan admin yang dapat dilihat pada Gambar 1. Kedua aktor dapat melakukan beberapa aktivitas sebagai berikut:

Bedasarkan Gambar 1, admin dan *user* dapat melakukan *login*, menambahkan alamat *email*, mengubah alamat *email*, menambahkan *password*, dan mengubah *password*. *User* juga dapat membuat akun atau *register*, membuat catatan transaksi, menambahkan kategori baru, dan menghapus kembali kategori yang sudah dibuat oleh *user*. Sedangkan, admin dapat mengambil seluruh data dari *user*, mengubah data, dan menghapus data.




Gambar 1. *Usecase diagram*

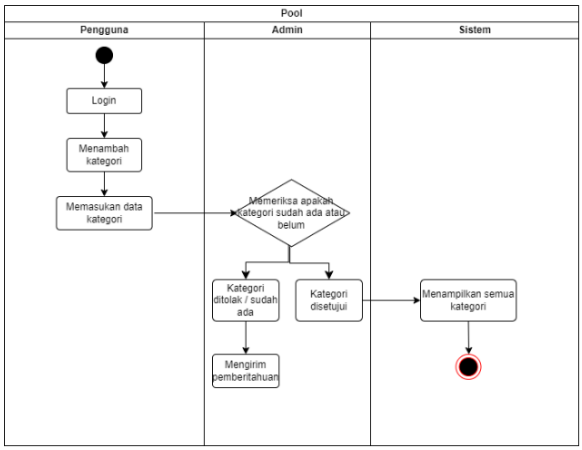
*Activity diagram* adalah salah satu bentuk pemodelan yang digunakan untuk memvisualisasikan aliran fungsionalitas suatu sistem. Berikut adalah *activity diagram* dalam melakukan transaksi:




Gambar 2. *Activity diagram* transaksi

*User* perlu melakukan proses *register* atau pembuatan akun terlebih dahulu jika belum memiliki akun. Selanjutnya, *user* dapat melakukan *login*. Lalu, *user* dapat melakukan tambah catatan transaksi. *User* perlu mengisi beberapa informasi yang diminta seperti jenis transaksi, waktu atau tanggal terjadinya transaksi, jumlah atau nominal uang, kategori transaksi, dan deskripsi atau keterangan pemakaian transaksi. Setelah itu, *user* dapat melakukan submit catatan transaksi dan sistem akan melakukan kalkulasi jumlah transaksi yang *user* masukkan. Admin akan melakukan pemeriksaan pada informasi yang dimasukkan oleh *user*. Jika sudah sesuai sistem akan menampilkan informasi transaksi serta hasil total transaksi kepda *user*.

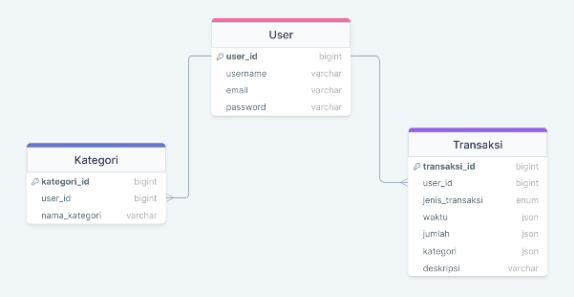
Aplikasi catatan keuangan ini memiliki beberapa pilihan kategori yang dapat digunakan oleh *user* untuk menggolongkan setiap transaksi yang dilakukan. Namun, *user* dapat menambahkan kategori baru sesuai yang diinginkan user dan dapat menghapusnya kembali jika sudah tidak dibutuhkan. Berikut adalah *activity diagram* kategori:




Gambar 3. *Activity diagram* kategori

Setelah melakukan proses *login*, *user* dapat menggunakan fitur tambah kategori dan memasukan nama kategori yang diingikan. Lalu admin akan memeriksa apakah kategori baru itu sudah tersedia atau belum. Jika sudah tersedia, sistem akan menampilkan pemberitahuan menolak kategori baru. Sedangkan belum tersedia, sistem akan menampilkan semua pilihan kategori termasuk kategori baru yang dibuat oleh *user*.

Berikut adalah *Entity Relationship Diagram* (ERD) dari aplikasi pencatatan keuangan.




Gambar 4. *Entity Relationship Diagram*

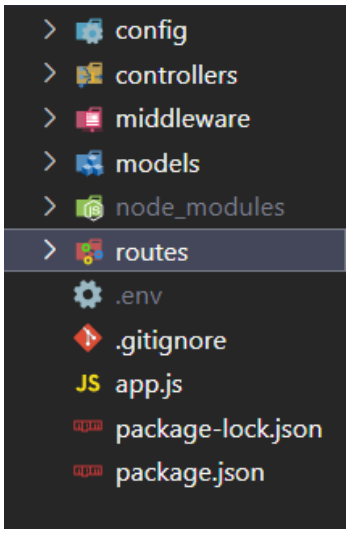
### 3.2 Rancangan Arsitektur Sistem


Gambar 5. Alur kerja api

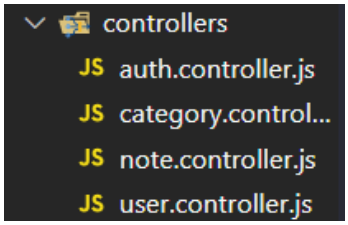
Pada Gambar 5, API bekerja ketika *client* membutuhkan data atau informasi dengan melakukan *request* kepada *server*, lalu *server* menerima dan memberikan *response* dalam format *Javascript Object Nation* (JSON) tanpa mengubah data aslinya. Metode *GET, POST, PUT,* dan *DELETE* adalah bentuk *request* yang dapat dilakukan oleh *client.*

Struktur *file coding* dalam rancang bangun *REST API* sangat penting diperhatikan untuk memudahkan tim dalam pengembangan dan menghindari konflik yang terjadi dalam *coding* saat kolaborasi tim. Dalam penelitian ini, terdapat susunan folder dibuat menggunakan konsep *design* arsitektur *MVC (Model, Control, View)* yang fungsi masing-masing dalam proses *development*.

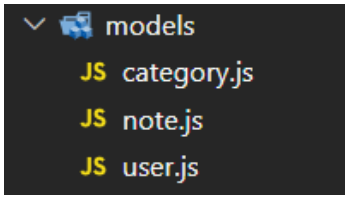
Berikut tampilan dari susunan folder pada aplikasi pencatatan keuangan:



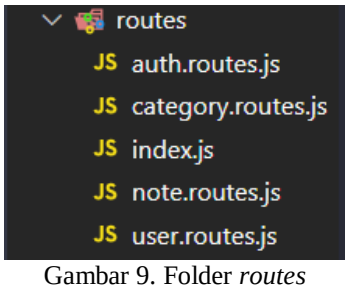

Gambar 6. Susunan folder



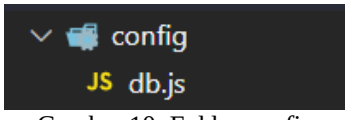

Gambar 7. Folder *controllers*




Gambar 8. Folder *models*




Gambar 9. Folder *routes*




Gambar 10. Folder *config*


Gambar 11. Folder *middleware*

Pada Gambar 7, terdapat folder *controllers* yang menyimpan beberapa file. *Controllers* digunakan untuk menyimpan semua logika yang dibuat untuk setiap fungsionalitas sistem. Pada Gambar 8, *models* digunakan untuk menyimpan data atau informasi yang dibutuhkan oleh sistem aplikasi pencatatan keuangan. Pada Gambar 9, *routes* digunakan untuk menyimpan file yang berisi seluruh *rute* atau *endpoint* dari aplikasi web. Pada Gambar 10, *config* digunakan untuk konfigurasi koneksi ke *database*. Terakhir Gambar 11, *middleware* digunakan untuk menyimpan fungsi-funsi yang berjalan untuk memgirimkan *respons* dari *server* ke *client.*

Pendekatan *mongoose* diimplementasikan dalam beberapa folder yaitu *config* dan *models.* Dalam *config*, pendekatan *mongoose* digunakan untuk membangun koneksi antara aplikasi backend dan *Mongoose database*. Pada Gambar 10, folder *config* menyimpan file *db.js* di dalamnya. Berikut tampilan dari file *db.js*:


Gambar 12. File *db.js*

Rancangan *REST API* memiliki beberapa informasi yang dibutuhkan dari *client* dan semua data atau informasi yang dibutuhkan tersimpan dalam folder *models.* Selain itu, *mongoose* juga digunakan pada *models*. Berikut tampilan dari masing-masing model:


Gambar 13. Model *user*

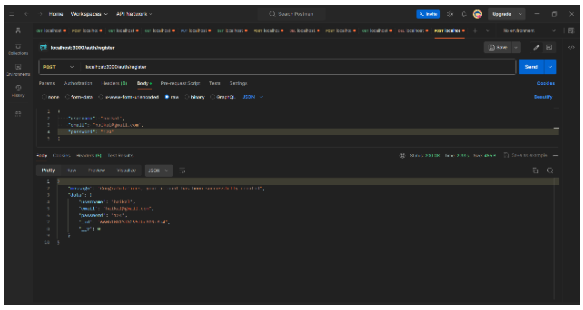

Gambar 14. Model *note*


Gambar 15. Model *category*

### 3.3 Rancangan *API*

Langkah akhir dalam penelitian ini adalah merilis *API* yang telah selesai. Namun, tim *backend* perlu melakukan pengecekan terhadap *route* pada setiap *API* untuk memastikan apakah *API* berjalan dengan baik dan sesuai . Proses pengecekan *route* dilakukan dengan menggunakan *tools POSTMAN. Output* atau hasil *API* akan tampil dalam bentuk *Object* atau *JSON.*

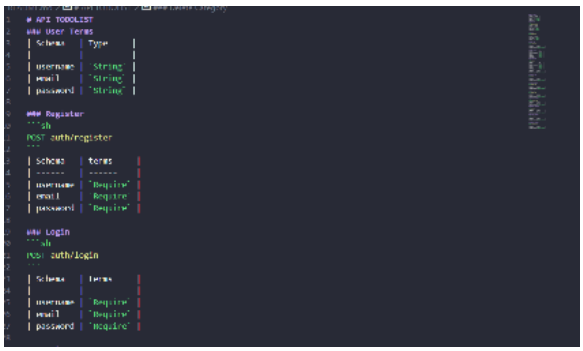
Tampilan data *API* dalam bentuk *JSON* menggunakan *POSTMAN* adalah sebagai berikut:



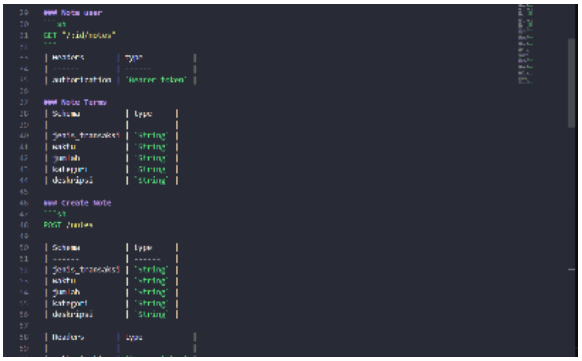
Gambar 16. *Output data*

## 4. HASIL DAN PEMBAHASAN

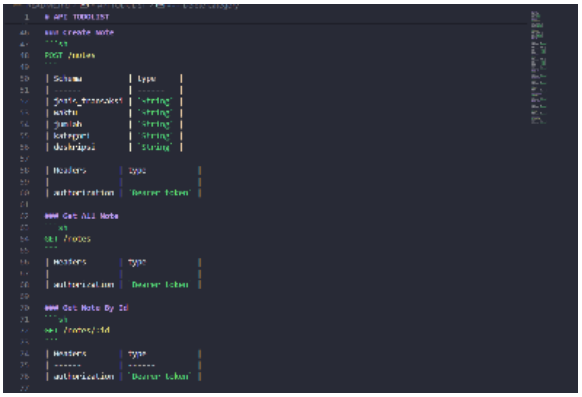
Penelitian ini menghasilkan beberapa *endpoint* yang melewati proses pengecekan menggunakan *POTSMAN*. *Endpoint* yang digunakan dapat dilihat sebagai berikut:



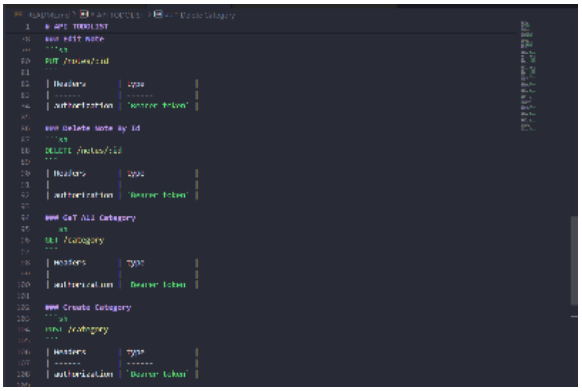
Gambar 17. *Endpoint user*



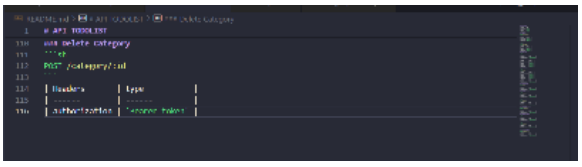
Gambar 18. *Endpoint note*



Gambar 19. Lanjutan *endpoint note*

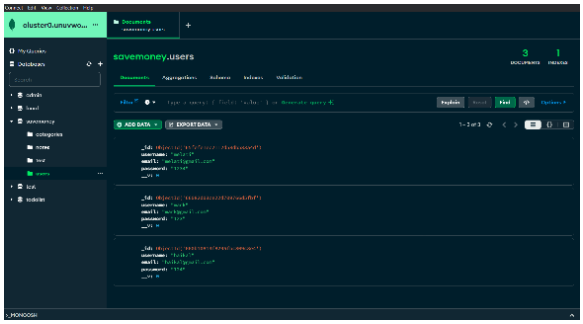


Gambar 20. *Endpoint category*



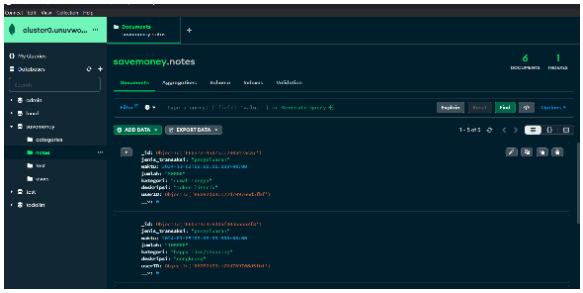
Gambar 21. Lanjutan *endpoint category*

Berikut beberapa hasil pengecekan *API* yang berhasil masuk dalam *database Mongodb*:



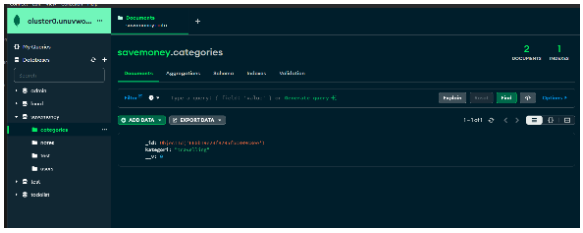
Gambar 22. *User database*

Pada Gambar 22, data pengguna yang telah melakukan *login* akan tersimpan dalam *user database.*



Gambar 23. Catatan transaksi *database*

Pada Gambar 23, semua catatan transaksi yang telah dibuat oleh pengguna akan tersimpan dalam Catatan transaksi *database.*



Gambar 24. *Category database*

Pada Gambar 24, kategori yang dibuat oleh pengguna akan tersimpan dalam *Category database.*

## 5. KESIMPULAN DAN SARAN

Dari hasil penelitian yang dilakukan, sistem aplikasi pencatatan keuangan berbasis web dengan penerapan *mongoose* dan metode *Agile Software Development* terbukti berhasil untuk membuat rancang bangun *REST API* aplikasi pencatatan keuangan. Dengan adanya *REST API*, *user* tidak perlu khawatir lagi mengenai kehilangan data atau informasi keuangan yang dibuat. Melalui metode penelitian dapat disimpulkan bahwa sistem berhasil mengatasi masalah kekhawatiran individu mengenai mudahnya kehilangan data atau informasi transaksi. Konseptual diagram atau UML juga membantu dalam pengembangan sistem secara terstruktur. Saran kedepannya adalah lanjutkan pengembangan sistem dan lakukan pengecekan secara berkala agar pemeliharaan tetap berjalan optimal.

**DAFTAR PUSTAKA**

[1] M. Darwis, D. Apriani, D. Islamiyati, and R. Agam, "Pengembangan Aplikasi MyNeeds Berorientasi Objek Untuk Memantau Keuangan Mahasiswa," 2023.

[2] F. Fernando, G. Hoendarto, and T. Willay, "APLIKASI MANAJEMEN KEUANGAN BERBASIS ANDROID TERINTEGRASI SISTEM PENGINGAT DAN PENCATAT KEUANGAN."

[3] "292215-sistem-pencatatan-dan-pengolahan-keuanga-3e25d2e9".

[4] "1474-Article Text-4030-1-10-20220219".

[5] I. R. D. Muhammad and I. V. Paputungan, "Development of Backend Server Based on REST API Architecture in E-Wallet Transfer System," *J. Sains, Nalar, dan Apl. Teknol. Inf.*, vol. 3, no. 2, pp. 79–87, Jan. 2024, doi: 10.20885/snati.v3.i2.35.

[6] I. Widjaja, "Rancang Bangun Aplikasi Manajemen Keuangan Rt (Rukun Tetangga) Berbasis Android," *J. Instrumentasi dan Teknol. Inform.*, vol. 2, no. 2, pp. 102–112, 2021, [Online]. Available: https://jurnal.poltek-gt.ac.id/index.php/jiti/102

[7] E. Nurhayati and A. Agussalim, "Rancang Bangun Back-end API pada Aplikasi Mobile AyamHub Menggunakan Framework Node JS Express," *J. Sist. dan Teknol. Inf.*, vol. 11, no. 3, p. 524, 2023, doi: 10.26418/justin.v11i3.66823.

[8] M. A. Ramdhan, A. Fauzi, A. S. Nugroho, M. A. Ramdhan, A. Fauzi, and A. S. Nugroho, "Implementasi Back-end Sistem Otomasi Pendataan Mahasiswa pada Food Truck Undip Menggunakan Framework Express.js," *J. Tek. Komput.*, vol. 2, no. 2, pp. 115–122, 2023, doi: 10.14710/jtk.v2i2.38527.

[9] I. Kurniawan, Humaira, and F. Rozi, "REST API Using NodeJS on Android-Based Electronic Service Transaction Application," *Sci. J. Inf. Syst. Technol.*, vol. 1, no. 4, pp. 127–132, 2020.

[10] R. Laipaka, "Penerapan Web Service JSON pada Backend-Developer Summary Report Executive Menggunakan Arsitektur MVC CodeIgniter," *Semin. Nas. Sist. Inf. dan Teknol. Inf.*, vol. 12, pp. 871–876, 2018.