# Using Supabase in Node.js

Supabase is a great open-source alternative to Firebase, based on PostgreSQL. It offers authentication, real-time database, storage, and much more. In this guide, we will see how to integrate it into a Node.js application.

## Installation

To get started, install the Supabase client using npm:

```
npm install @supabase/supabase-js
```

## Configuration

Once the package is installed, configure the Supabase client with your URL and API key. You can find this information in the Supabase dashboard.

```
const { createClient } = require('@supabase/supabase-
js');

const SUPABASE_URL = 'https://your-project-
url.supabase.co';
const SUPABASE_KEY = 'your-anon-key';

const supabase = createClient(SUPABASE_URL,
SUPABASE_KEY);
```

## Basic Operations

### Insert Data

To insert data into a table, use the `insert` method:

```
async function insertData() {
  const { data, error } = await supabase
    .from('users')
    .insert([
      { name: 'Mario Rossi', email:
'mario@example.com' }
    ]);

  if (error) console.error(error);
  else console.log(data);
}

insertData();
```

## Retrieve Data

To fetch data from a table, use the `select` method:

```
async function getData() {
  const { data, error } = await supabase
    .from('users')
    .select();

  if (error) console.error(error);
  else console.log(data);
}

getData();
```

## Update Data

You can update data using the `update` method:

```
async function updateData() {
  const { data, error } = await supabase
    .from('users')
    .update({ name: 'Luigi Bianchi' })
    .eq('email', 'mario@example.com');

  if (error) console.error(error);
  else console.log(data);
}

updateData();
```

### Delete Data

To delete a record, use the `delete` method:

```
async function deleteData() {
  const { data, error } = await supabase
    .from('users')
    .delete()
    .eq('email', 'mario@example.com');

  if (error) console.error(error);
  else console.log(data);
}

deleteData();
```

## Authentication

Supabase also offers authentication with email and password. Here’s how to register a user:

```
async function signUp() {
  const { user, error } = await
supabase.auth.signUp({
    email: 'test@example.com',
    password: 'password123'
  });

  if (error) console.error(error);
  else console.log(user);
}

signUp();
```

And to log in:

```
async function signIn() {
  const { user, error } = await
supabase.auth.signInWithPassword({
    email: 'test@example.com',
    password: 'password123'
  });

  if (error) console.error(error);
  else console.log(user);
}

signIn();
```

## Conclusion

Supabase greatly simplifies database and authentication management in a Node.js application. Thanks to its intuitive API, you can easily integrate advanced features without managing complex servers.