Praktikum 3

DAFTAR ISI

TUJUAN PRAKTIKUM 1
SOFTWARE YANG DIBUTUHKAN 1
TIPS SEBELUM MEMULAI 2
MEMBUAT APLIKASI DAN MENGATUR KONEKSI PADA HEROKU 4
MEMBANGUN REST API UNTUK MENAMPILKAN DATA 13
MENGAKSES GAMBAR PADA REST API 17
MEMBANGUN REST API UNTUK PROSES CREATE, READ, UPDATE, DAN DELETE 19
TUGAS 32

TUJUAN PRAKTIKUM

- Mahasiswa mampu membuat rest api
- Mahasiswa

SOFTWARE YANG DIBUTUHKAN

- Node.js (https://nodejs.org/)
- Visual code (https://code.visualstudio.com/)
- Xampp (https://www.apachefriends.org/index.html)
- Postman (https://www.getpostman.com/)
- Git (https://git-scm.com/)
- Heroku (https://www.heroku.com/)

TIPS SEBELUM MEMULAI

1. Contoh kode:

   Kode 1

```
// cara mengakses gambar http://polibatam.ac.id/img/perpustakaan.png
```

   Kode 2

```
// cara mengakses gambar
http://polibatam.ac.id/img/perpustakaan.png
```

   Kode 3

```
// cara mengakses gambar
//http://polibatam.ac.id/img/perpustakaan.png
```

   Apakah terdapat perbedaan kode 1, kode 2, dan kode 3?

   Symbol // digunakan untuk komentar perbaris.

   Jika menemukan kode 2, maka jadikan 1 baris (bukan 2 baris).

2. Contoh kode:

```
sql.query("CREATE TABLE books (id int NOT NULL AUTO_INCREMENT
title VARCHAR(255) NOT NULL, description VARCHAR(255), images
VARCHAR(255), created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY(id))"
```

   Apakah anda akan menuliskan ulang kode di atas menjadi 1 baris atau 4 baris ?

   Kode di atas mestinya hanya 1 baris.

   Jika diminta untuk menulis menjadi 4 buah baris maka akan menjadi:

```
sql.query("CREATE TABLE books (id int NOT NULL AUTO_INCREMENT, "
    + "title VARCHAR(255) NOT NULL, description VARCHAR(255), "
    + "images VARCHAR(255), created_at TIMESTAMP "
    + "DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY(id))");
```

3. Contoh kode:

   Pada html:

   1 <ion-button>KLIK SAYA</ion-button>

   2 </ion-button>KLIK SAYA</ion-button>

   3 <ionbutton>>KLIK SAYA<ionbutton>

   4 <ion-button>KLIK SAYA

   5 <ion-button KLIK SAYA><ion-button>

6 <ion-bu

tton> KLIK SAYA<ion-button>

1,2,3,4,5, dan 6.

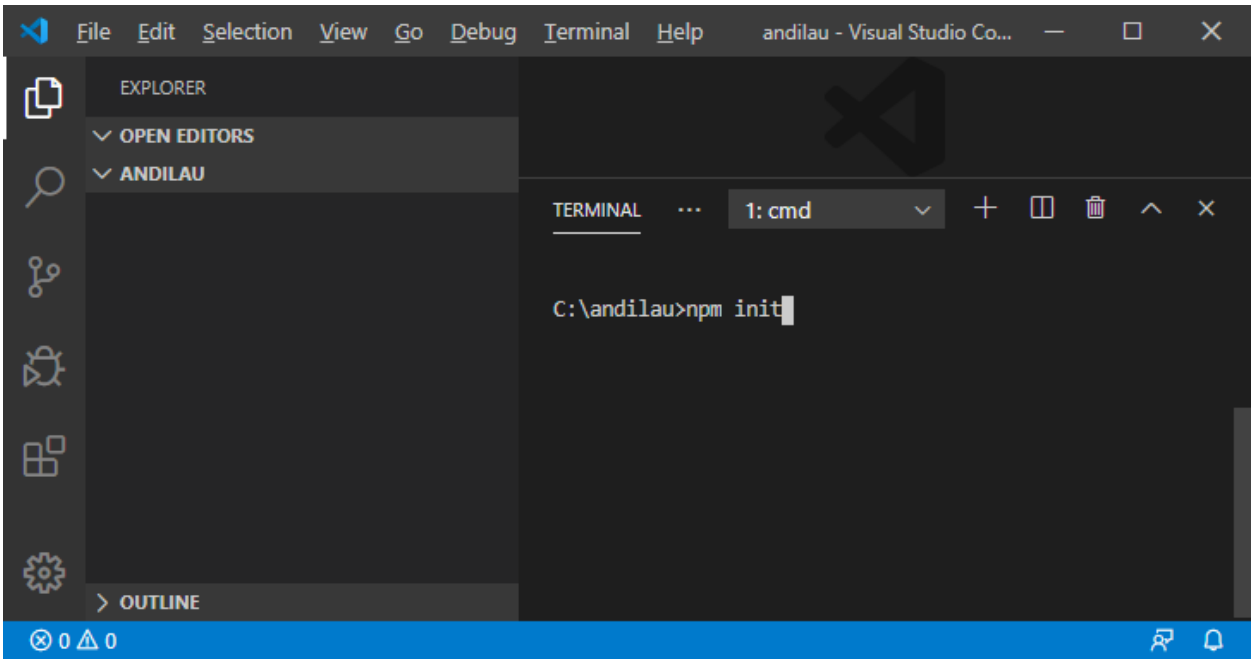
Nomor berapa kode yang benar?

Baris 1 sesuai dengan panduan pada ionicframework.com, jika anda tidak sengaja menulis kode seperti 2,3,4,5 dan 6. Silahkan cek kembali slide teori atau ionicframework.com

4. Jika mengalami error, namun error tidak muncul, perhatikan kembali apa yang tidak sengaja anda rubah sebelumnya dan manfaatkan developer console (cek pada slide)
5. Jika anda memanggil service cart, namun service cart belum pernah dibuat. Apakah aplikasi akan berjalan normal? Cek pada slide untuk membuat service.
6. Dalam praktikum ini, manfaatkan console.log() untuk menampilkan informasi pada console. Jika consolenya pada heroku.com, jalankan perintah `heroku logs --tails`

MEMBUAT APLIKASI DAN MENGATUR KONEKSI PADA HEROKU

1. Buka visual code > open folder pada folder yang anda inginkan.
   - `Contoh: D:\andilau\`
2. Buka terminal pada visual code. Menu View > terminal
3. Jalankan perintah pada terminal
   - `npm init`



Perintah `npm init` untuk menginisialisasi project nodejs.

Selanjutnya ketika mengisi informasi mengenai project, hanya diisi pada bagian entry point, Ketik: server.js

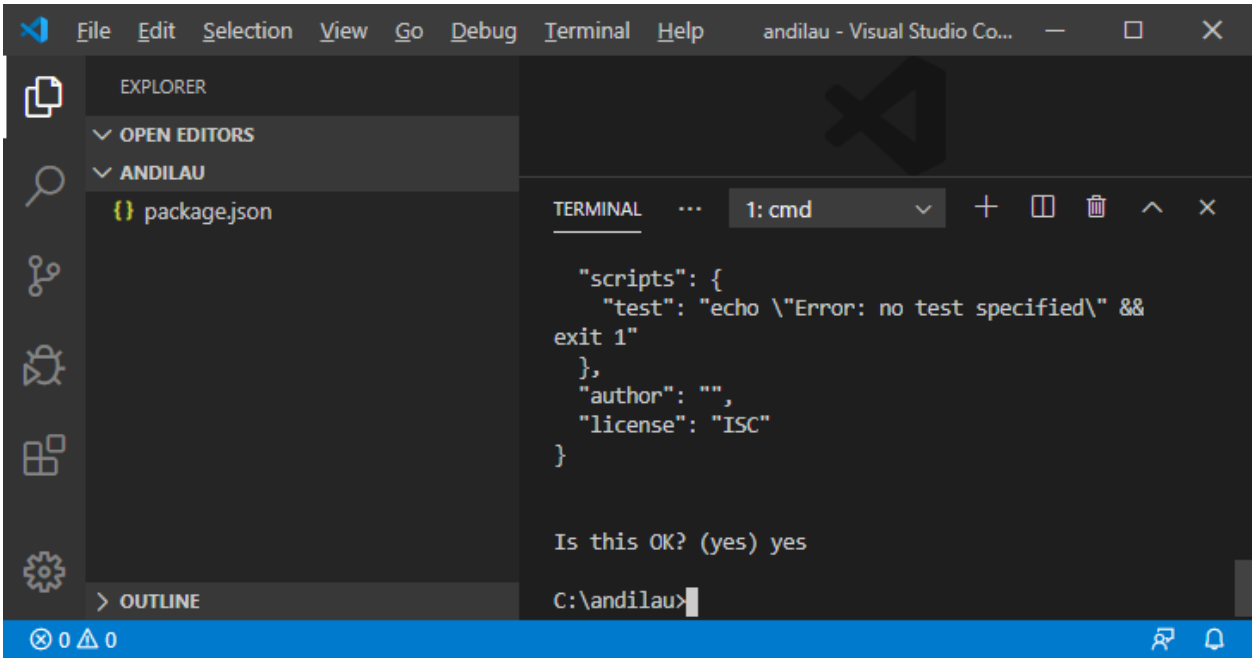
Selebihnya kosongkan (langsung di enter).

Pada bagian akhir, ketik perintah yes untuk setuju.

```
• Package                    :
• Version (1.0.0)            :
• Description                :
• Entry point: (index.js): server.js
• Test command               :
• Git repository             :
```

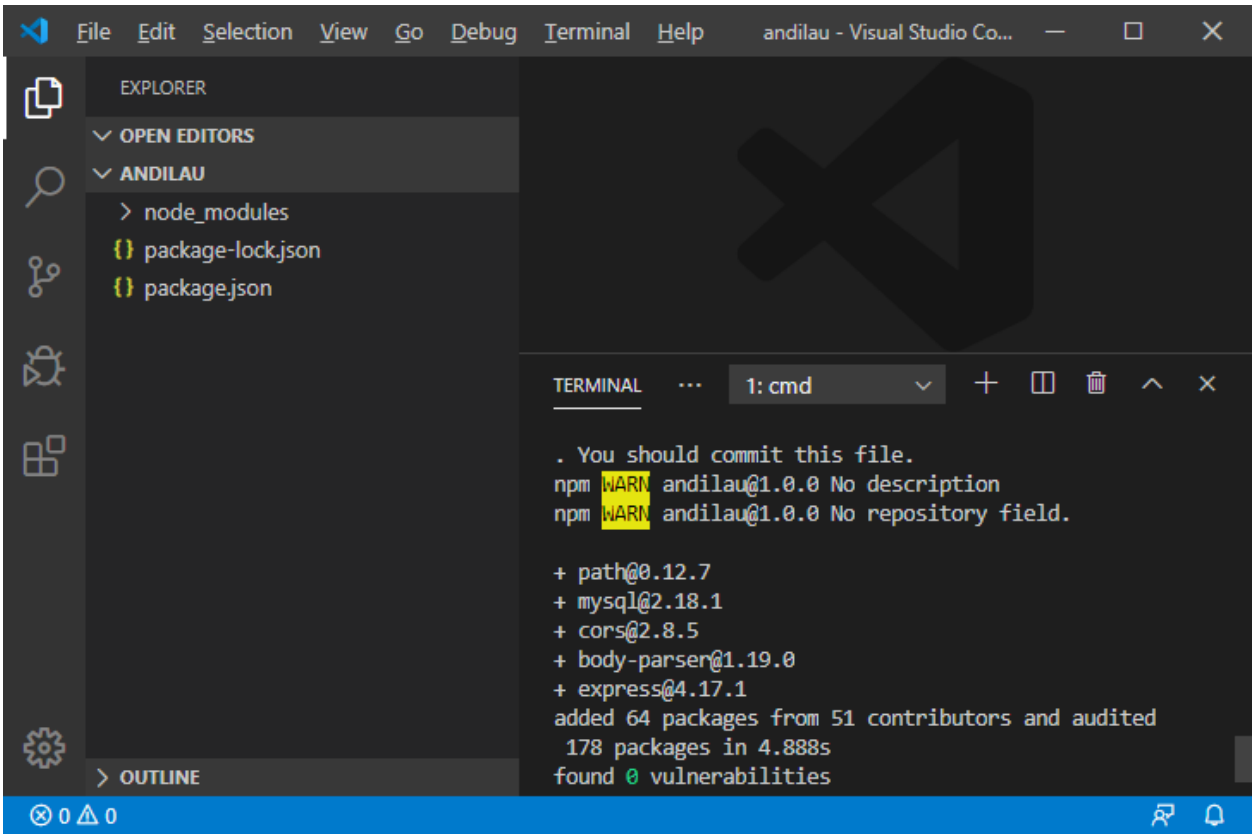
```
- Keywords                  :
- Author                    :
- License (ISC)             :

- Is this OK? (yes)         : yes
```

4. Hasilnya

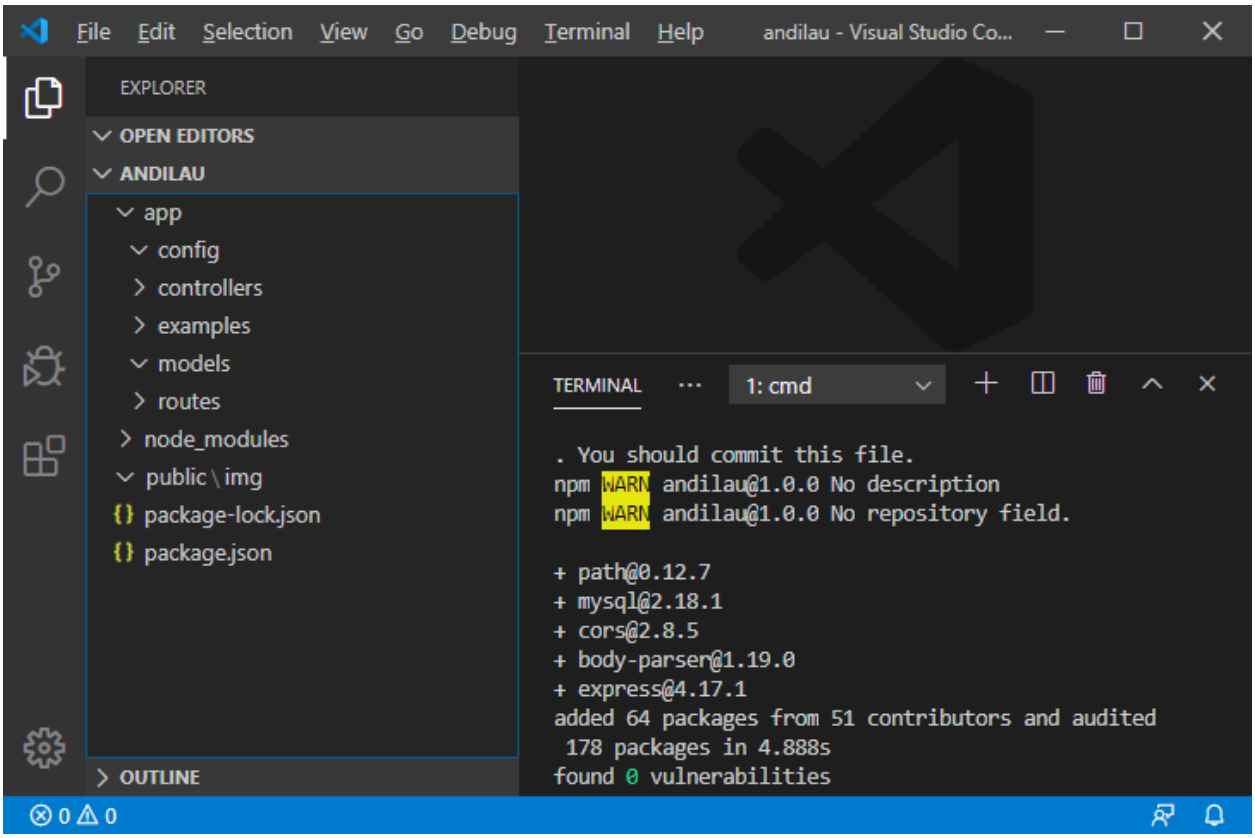


5. Jalankan perintah berikut untuk menginstal library yang dibutuhkan

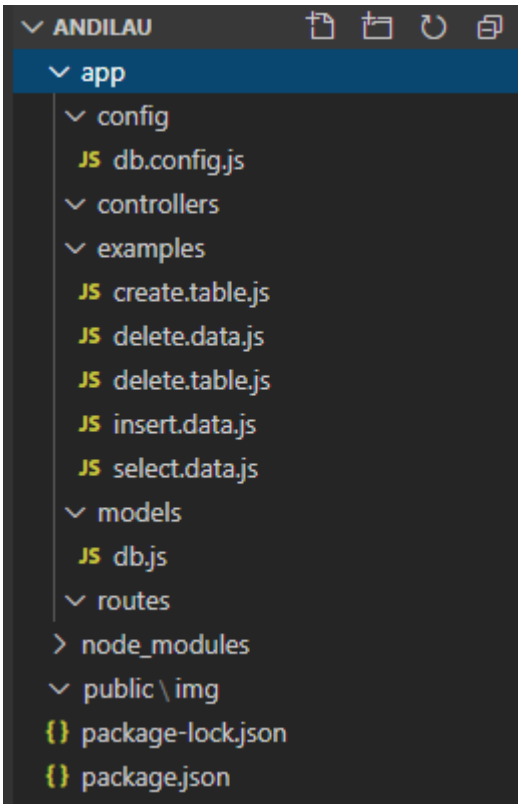
   - `npm install express mysql body-parser path cors --save`



6. Buatlah folder app, lalu didalamnya buatlah folder config, controllers, examples, models, dan routes.
7. Buatlah folder public, lalu didalamnya buatlah folder img



8. Selanjutnya anda akan diminta untuk membuat beberapa file seperti dibawah ini:



9. Didalam folder config, buat file baru, beri nama db.config.js

```
//app/config/db.config.js
module.exports = {
  HOST: "xxx",
  USER: "xxx",
  PASSWORD: "xxx",
  DATABASE: "xxx"
};
```

10. Didalam folder models, buatlah file baru, beri nama db.js.

```
//app/models/db.js
const mysql = require("mysql");
const dbConfig = require("../config/db.config.js");
var connection = mysql.createPool({
  host: dbConfig.HOST,
  user: dbConfig.USER,
  password: dbConfig.PASSWORD,
  database: dbConfig.DATABASE
});
module.exports = connection;
```

11. Didalam folder examples, buatlah file baru, beri nama create.table.js

```
//app/examples/create.table.js
const sql = require("../models/db");
sql.query("CREATE TABLE books (id int NOT NULL AUTO_INCREMENT, "
    + "title VARCHAR(255) NOT NULL, description VARCHAR(255), "
    + "images VARCHAR(255), created_at TIMESTAMP "
    + "DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY(id))",
    (err, res) => {
        if (err) {
            console.log(err);
        } else {
            console.log("Table berhasil dibuat");
        }
    });
```

Perhatikan spasi pada query

12. Didalam folder examples, buatlah file baru, beri nama delete.table.js

```
//app/examples/delete.table.js
const sql = require("../models/db");
sql.query("DROP TABLE books", (err, res) => {
  if (err) {
```

```
    console.log(err);
  } else {
    console.log("Table berhasil dihapus");
  }
});
```

13. Didalam folder examples, buatlah file baru, beri nama select.data.js

```
//app/examples/select.data.js
const sql = require("../models/db");
sql.query("SELECT * FROM books", (err, res) => {
  if (err) {
    console.log("error: ", err);
  } else {
    console.log("result: ", res);
  }
});
```

14. Didalam folder examples, buatlah file baru, beri nama insert.table.js

```
//app/examples/insert.data.js
const sql = require("../models/db");
sql.query("INSERT INTO books VALUES (NULL, 'title', 'desc', 'per
pustakaan.png', current_timestamp());", (err, res) => {
  if (err) {
    console.log(err);
  } else {
    console.log("Data berhasil ditambahkan: " + res.affectedRows
);
  }
});
```

15. Didalam folder examples, buatlah file baru, beri nama delete.data.js

```
//app/examples/delete.data.js
const sql = require("../models/db");
sql.query("DELETE FROM BOOKS WHERE id = '1'", (err, res) => {
  if (err) {
    console.log(err);
  } else {
    console.log("Data berhasil dihapus: " + res.affectedRows);
  }
});
```

16. Pada terminal, jalankan perintah untuk melakukan cek versi heroku

    - `heroku –v`

    ```
    C:\andilau>heroku -v
    heroku/7.35.1 win32-x64 node-v13.6.0

    C:\andilau>
    ```

    Jika tidak muncul versi heroku (heroku belum di install), maka jalankan perintah:

    - `npm install –g heroku`

    lebih lanjut: https://devcenter.heroku.com/articles/heroku-cli

17. Jika berhasil di instal, maka silahkan cek kembali versi heroku
18. Pada terminal, login heroku dengan perintah

    - `Heroku login`

    Jika belum memiliki account, silahkan daftar
19. Membuat git repository

    - `git init`
    - `git add .`
    - `git commit –m "commit awal"`

20. Membuat aplikasi heroku dan verifikasi

    - `heroku create`
    - `git remote –v`

    ```
    C:\andilau>heroku create
    Creating app... done, ⬢ hidden-earth-26826
    https://hidden-earth-26826.herokuapp.com/ | https://git.heroku.com/hidden-earth-26826.git

    C:\andilau>git remote -v
    heroku  https://git.heroku.com/hidden-earth-26826.git (fetch)
    heroku  https://git.heroku.com/hidden-earth-26826.git (push)
    ```

21. Menambahkan database cleardb pada heroku

    - `heroku addons:create cleardb:ignite`

```
C:\andilau>heroku addons:create cleardb:ignite
Creating cleardb:ignite on ⬢ hidden-earth-26826... free
Created cleardb-regular-30737 as CLEARDB_DATABASE_URL
Use heroku addons:docs cleardb to view documentation
```

22. Mengatur koneksi mysql

    - `heroku config | grep CLEARDB_DATABASE_URL`
    - `heroku config | findstr CLEARDB_DATABASE_URL`

```
C:\andilau>heroku config
=== hidden-earth-26826 Config Vars
CLEARDB_DATABASE_URL: mysql://bd130a55b5567b:9b2fc
1a5@us-cdbr-iron-east-04.cleardb.net/heroku_de0fa4
1b6e800f7?reconnect=true
```

```
C:\andilau>heroku config
=== hidden-earth-26826 Config Vars
CLEARDB_DATABASE_URL: mysql://bd13[redacted]67b:9[redacted]c
1a5@us-cdbr-iron-east-[redacted].cleardb.net/heroku_de[redacted]4
1b6e800f7?reconnect=true
```

    Dari url di atas, diterjembahkan menjadi

    mysql://username:password@host/database?reconnect=true

23. Buka file app/config/db.config.js
24. Kemudian modifikasi sesuai config di atas

    Contoh:

```
JS db.config.js ×
app > config > JS db.config.js > ...
2   module.exports = {
3       HOST: "us-cdbr-iron-east-xx.cleardb.net",
4       USER: "bd13xx7b",
5       PASSWORD: "9xxxc1a5",
6       DATABASE: "heroku_dexxx41b6e800f7"
7     };
```

25. Untuk menguji coba koneksi, arahkan terminal pada folder examples

```
C:\andilau>cd app/examples

C:\andilau\app\examples>
```

Lalu jalankan perintah

- `Node create.table.js`

```
C:\andilau\app\examples>node create.table.js
Table berhasil dibuat
^C
C:\andilau\app\examples>
```

Tekan ctrl + C untuk mematikan perintah

- `Node insert.data.js`

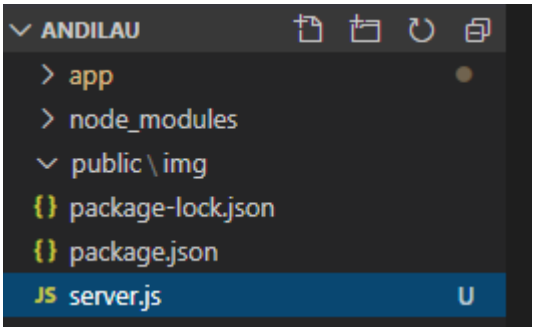
```
C:\andilau\app\examples>node insert.data.js
Data berhasil ditambahkan: 1
^C
```

- `Node select.data.js`

```
C:\andilau\app\examples>node select.data.js
result:  [
  RowDataPacket {
    id: 2,
    title: 'title',
    description: 'desc',
    images: 'perpustakaan.png',
    created_at: 2020-02-09T10:13:42.000Z
  }
]
```

MEMBANGUN REST API UNTUK MENAMPILKAN DATA

1. Buatlah sebuah file, beri nama server.js



2. Modifikasi file server.js

```
const express = require("express");
const bodyParser = require("body-parser");
var cors = require("cors")
var path = require('path');
const app = express();
// parse requests - application/json
app.use(bodyParser.json());
// parse requests - application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: true }));
//cors
var cor = cors();
app.use(cor);
//path
app.use(express.static(path.join(__dirname, "./public")));
app.get("/", (req, res) => {
  res.json({ message: "Selamat datang pada matakuliah
pemrograman perangkat bergerak" });
});
require("./app/routes/book.routes")(app);
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}.`);
});
```

3. Pada folder app/models, buatlah file baru, beri nama book.model.js

```
//app/models/book.models.js
const sql = require("./db.js");
const Book = function (book) {
  this.title = book.title;
  this.description = Book.description;
```

```
  this.images = Book.images;
};
//Mengambil semua data buku
Book.getAll = result => {
  sql.query("SELECT * FROM books", (err, res) => {
    if (err) {
      console.log("error: ", err);
      result(null, err);
      return;
    }
    console.log("result: ", res);
    result(null, res);
  });
};
module.exports = Book;
```

4. Pada folder controllers, buat file baru, beri nama book.controller.js

```
//app/controllers/book.controller.js
const Book = require("../models/book.model");
//Mengambil semua data buku
exports.findAll = (req, res) => {
  Book.getAll((err, data) => {
    if (err) {
      res.status(500).send({
        message:
          err.message || "Terjadi kesalahan"
      });
    } else res.send(data);
  });
};
```

5. Pada folder routes, buat file baru, beri nama book.routes.js

```
//app/routes/book.routes.js
module.exports = app => {
    const books = require("../controllers/book.controller");
    // cara mengakses gambar
http://polibatam.ac.id/img/perpustakaan.png
    // Mengambil semua data
    app.get("/api/books", books.findAll);
};
```

6. Push perubahan data pada git dan heroku (jalankan pada terminal)

```
git add .
```

```
git commit –m "menampilkan data"
git push heroku master
```

```
C:\andilau>git add .
```

```
C:\andilau>git commit -m "make it better"
[master c513c80] make it better
 5 files changed, 68 insertions(+), 4 deletions(-)
 create mode 100644 app/controllers/book.controller.js
 create mode 100644 app/models/book.model.js
 create mode 100644 app/routes/book.routes.js
 create mode 100644 server.js
```

```
C:\andilau>git push heroku master
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 4 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.97 KiB | 404.00 KiB/s, done.
Total 12 (delta 0), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
```

...

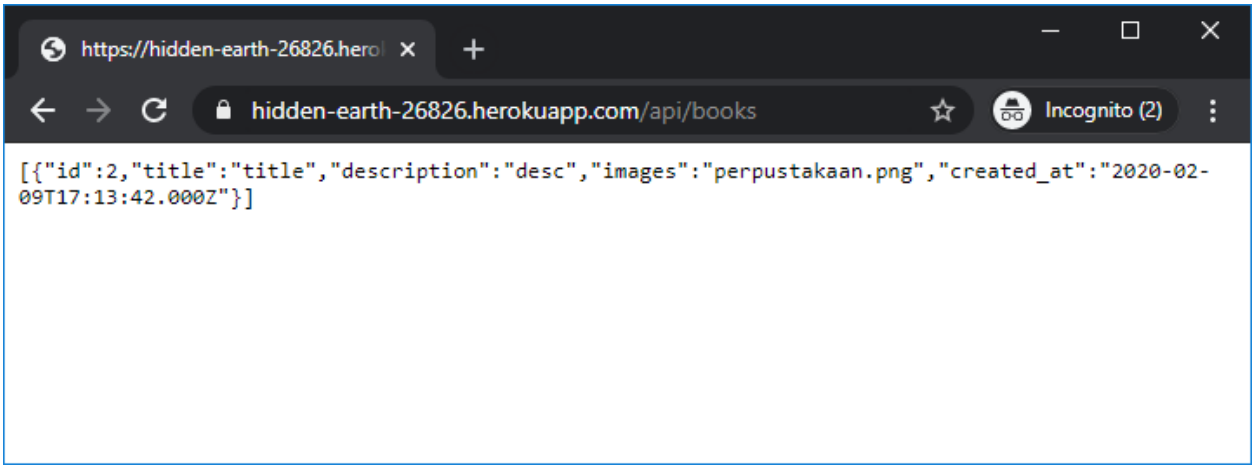
```
remote: Verifying deploy... done.
To https://git.heroku.com/hidden-earth-26826.git
   b6ac110..c513c80  master -> master
```

7. Untuk membuka aplikasi yang sudah di push, ketikkan perintah berikut

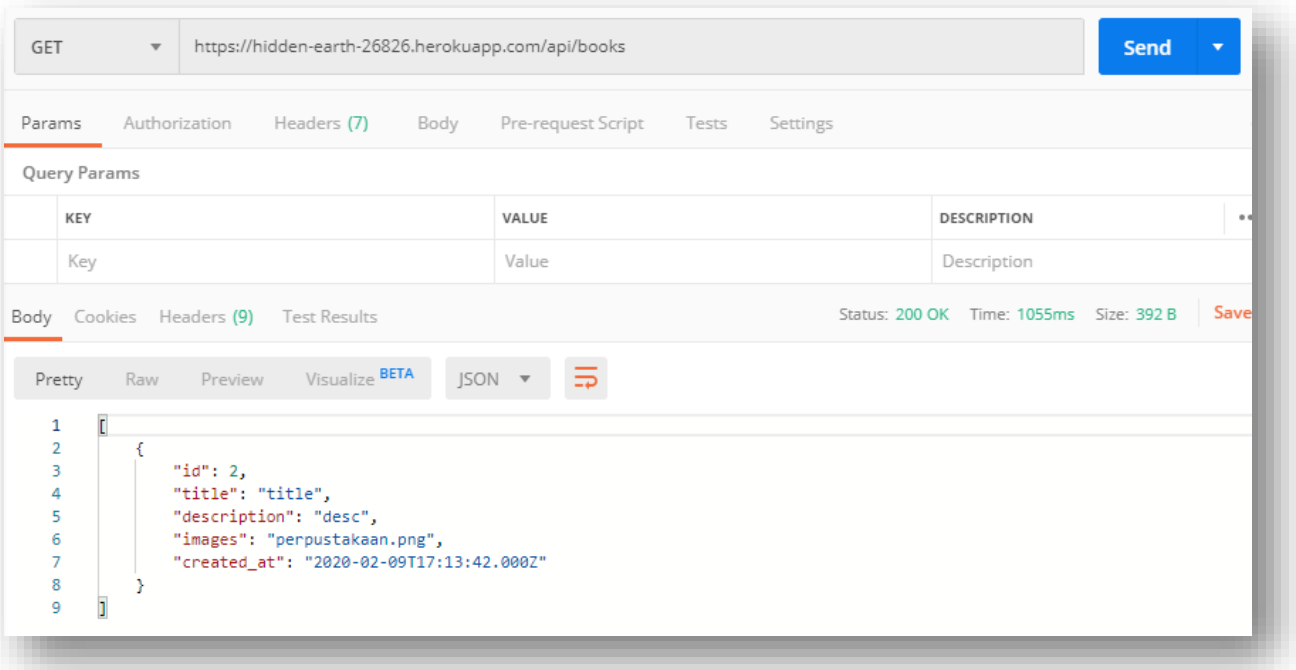
   - `heroku open api/books`

```
C:\andilau>heroku open /api/books
```

8. Hasilnya



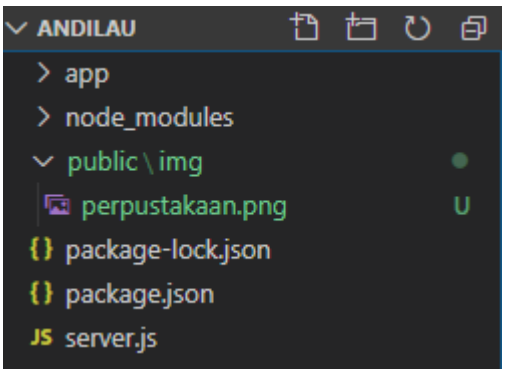
9. Atau pada aplikasi postman



MENGAKSES GAMBAR PADA REST API

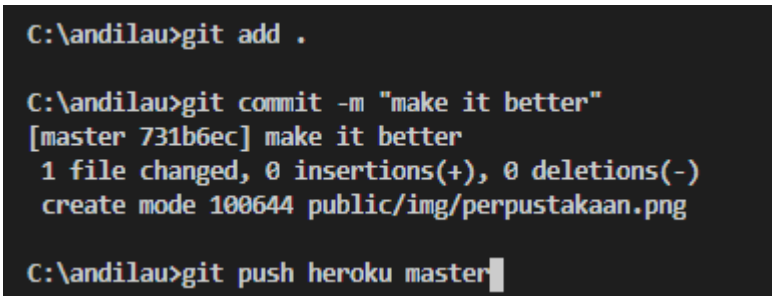
1. Letakkan gambar pada folder public/img

   Contoh nama gambar adalah perpustakaan.png



   push perubahan ke server

```
git add .
git commit –m "make it better"
git push heroku master
```

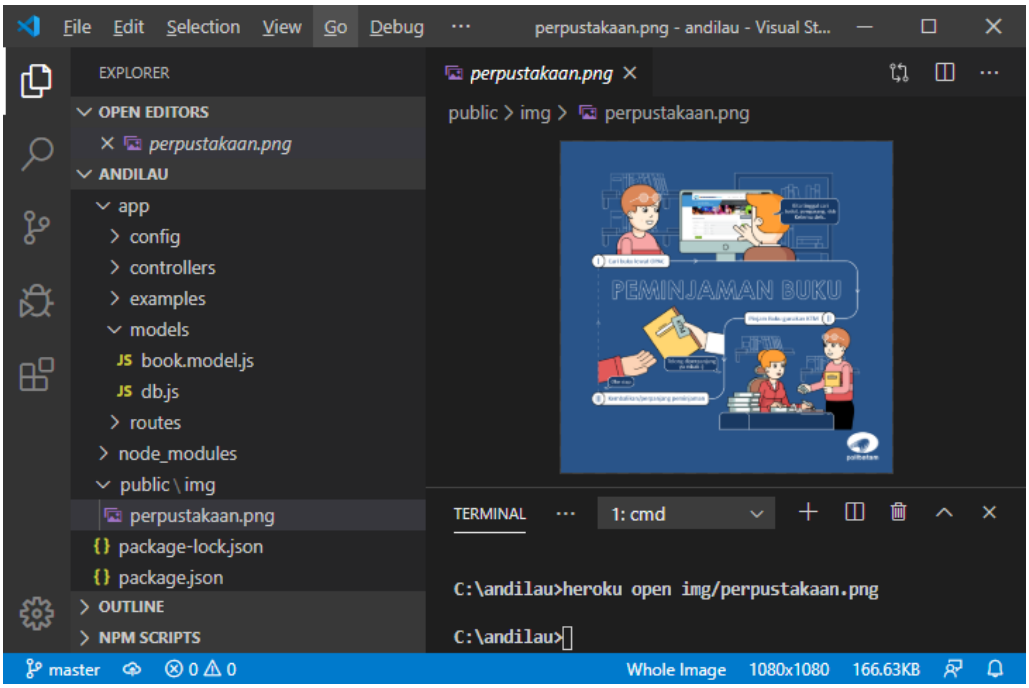


   Untuk mengakses, ketikkan:

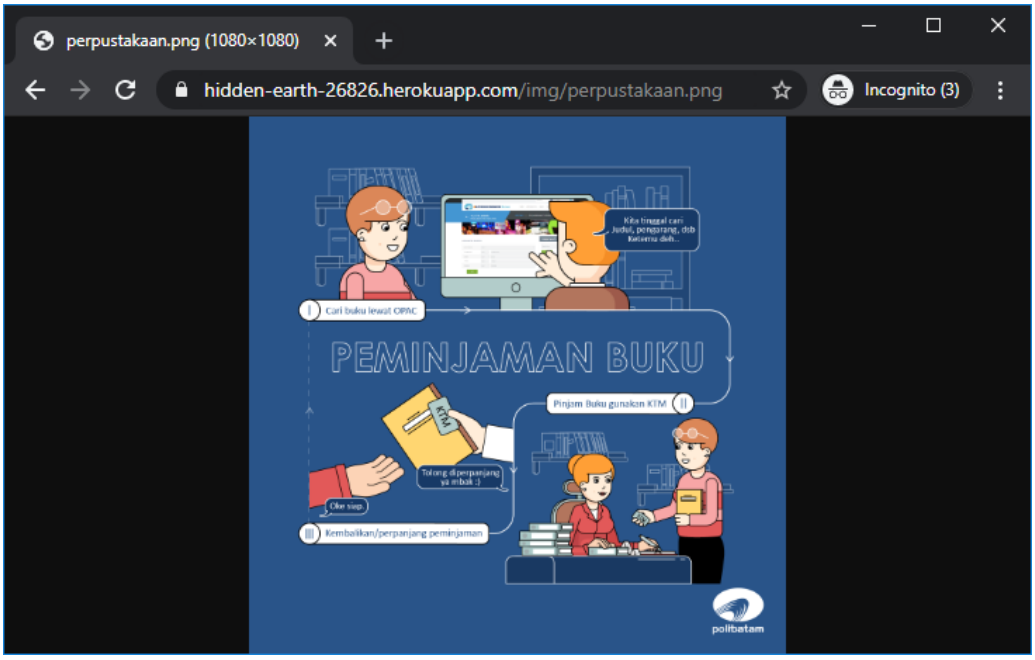
```
heroku open img/perpustakaan.png
```

2. atau langsung akses pada url heroku yang telah anda bangun lalu ditambahkan `img/perpustakaan.png`

contoh file perpustakaan.png pada project



hasil ketika mengakses pada heroku



MEMBANGUN REST API UNTUK PROSES CREATE, READ, UPDATE, DAN DELETE

1. Buka file books.model.js pada folder models
2. Modifikasi seperti dibawah ini

```
//app/models/book.models.js
const sql = require("./db.js");
const Book = function (book) {
  this.title = book.title;
  this.description = book.description;
  this.images = book.images;
};
//Mengambil semua data buku
Book.getAll = result => {
  sql.query("SELECT * FROM books", (err, res) => {
    if (err) {
      console.log("error: ", err);
      result(null, err);
      return;
    }
    console.log("result: ", res);
    result(null, res);
  });
};
// Mengambil buku yang memiliki id = BookId
Book.findById = (id, result) => {
  sql.query(`SELECT * FROM Books WHERE id = ${id}`, (err, res)
=> {
    if (err) {
      console.log("error: ", err);
      result(err, null);
      return;
    }
    if (res.length) {
      console.log(res[0]);
      result(null, res[0]);
      return;
    }
    result({ kind: "not_found" }, null);
  });
};
// Membuat data buku baru
Book.create = (newBook, result) => {
  console.log(newBook);
  sql.query("INSERT INTO books (title, description, images)
VALUES (?,?,?)",
```

```
    [newBook.title, newBook.description, newBook.images], (err,
res) => {
      if (err) {
        console.log("error: ", err);
        result(err, null);
        return;
      }
      console.log(res);
      console.log("buat buku: ", { id: res.insertId, ...newBook
});
      result(null, { id: res.insertId, ...newBook });
    });
};
// Mengupdate data buku yang memiliki id = id
Book.updateById = ( id, Book, result) => {
  sql.query(
    "UPDATE Books SET title = ?, description = ?, images = ?
WHERE id = ?",
    [Book.title, Book.description, Book.images, id],
    (err, res) => {
      if (err) {
        console.log("error: ", err);
        result(null, err);
        return;
      }
      if (res.affectedRows == 0) {
        result({ kind: "not_found" }, null);
        return;
      }
      console.log("update buku: ", { id: id, ...Book });
      result(null, { id: id, ...Book });
    }
  );
};
// Menghapus buku yang memiliki id = id
Book.remove = (id, result) => {
  sql.query("DELETE FROM Books WHERE id = ?", id, (err, res) =>
{
    if (err) {
      console.log("error: ", err);
      result(null, err);
      return;
    }
    if (res.affectedRows == 0) {
      // not found Book with the id
      result({ kind: "not_found" }, null);
      return;
    }
    console.log("hapus buku dengan id: ", id);
    result(null, res);
  });
```

```
};
// Menghapus semua buku
Book.removeAll = result => {
  sql.query("DELETE FROM Books", (err, res) => {
    if (err) {
      console.log("error: ", err);
      result(null, err);
      return;
    }
    console.log(`Menghapus ${res.affectedRows} buku`);
    result(null, res);
  });
};


module.exports = Book;
```

3. Buka file book.controller.js pada folder controllers
4. Modifikasi seperti dibawah ini

```
//app/controllers/book.controller.js
const Book = require("../models/book.model");
//Mengambil semua data buku
exports.findAll = (req, res) => {
  Book.getAll((err, data) => {
    if (err) {
      res.status(500).send({
        message:
          err.message || "Terjadi kesalahan"
      });
    } else {
      res.send(data);
    }
  });
};
// Mengambil buku yang memiliki id = id
exports.findOne = (req, res) => {
  Book.findById(req.params.id, (err, data) => {
    if (err) {
      if (err.kind === "not_found") {
        res.status(404).send({
          message: `Buku dengan id ${req.params.id} tidak
ditemukan`
        });
      } else {
        res.status(500).send({
```

```
          message: `Error ketika mengambil buku dengan id
${req.params.id}`
        });
      }
    } else {
      res.send(data);
    }
  });
};
// Membuat data buku baru
exports.create = (req, res) => {
  if (!req.body) {
    res.status(400).send({
      message: "Content tidak boleh kosong"
    });
  }
  const book = new Book({
    title: req.body.title,
    description: req.body.description,
    images: req.body.images
  });
  Book.create(book, (err, data) => {
    if (err) {
      res.status(500).send({
        message:
          err.message || "Terjadi kesalahan"
      });
    }
    else {
      res.send(data);
    }
  });
};
// Mengupdate data buku yang memiliki id = id
exports.update = (req, res) => {
  if (!req.body) {
    res.status(400).send({
      message: "Content tidak boleh kosong"
    });
  }
  Book.updateById(
    req.params.id,
    new Book(req.body), (err, data) => {
      if (err) {
        if (err.kind === "not_found") {
          res.status(404).send({
            message: `Buku dengan id ${req.params.id} tidak
ditemukan`
          });
        } else {
          res.status(500).send({
```

```
            message: `Error ketika mengupdate buku dengan id
${req.params.id}`
          });
        }
      } else {
        res.send(data);
      }
    }
  );
};
// Menghapus buku yang memiliki id = id
exports.delete = (req, res) => {
  Book.remove(req.params.id, (err, data) => {
    if (err) {
      if (err.kind === "not_found") {
        res.status(404).send({
          message: `Buku dengan id ${req.params.id} tidak
ditemukan`
        });
      } else {
        res.status(500).send({
          message: `Error ketika menghapus buku dengan id
${req.params.id}`
        });
      }
    } else res.send({ message: `Berhasil menghapus data buku!`
});
  });
};
// Menghapus semua buku
exports.deleteAll = (req, res) => {
  Book.removeAll((err, data) => {
    if (err) {

      res.status(500).send({
        message:
          err.message || "Terjadi kesalahan"
      });
    }
    else {
      res.send({ message: `Berhasil menghapus seluruh data
buku!` });
    }
  });
};
```

5. Buka file book.routes.js pada folder routes
6. Modifikasi seperti dibawah ini

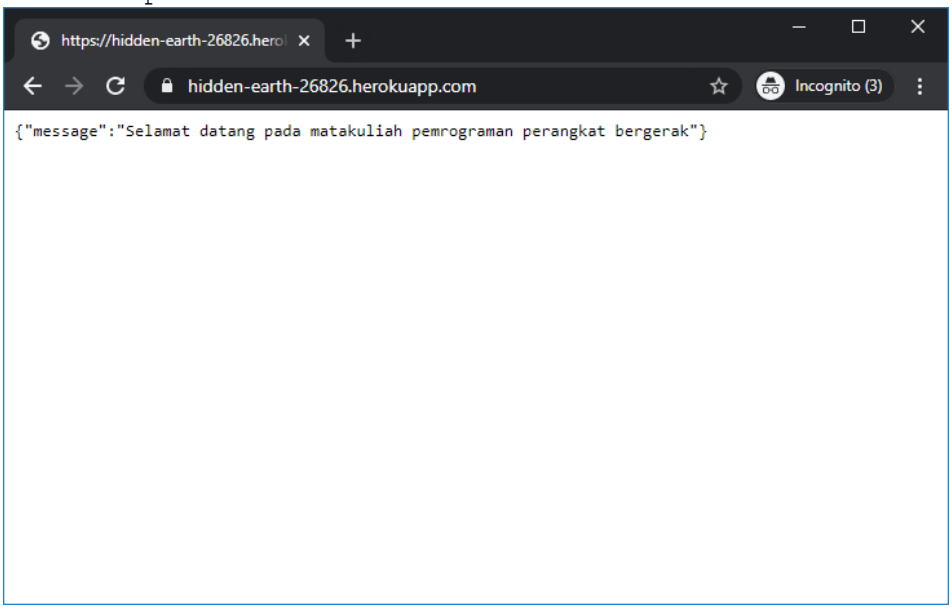
```
//app/routes/book.routes.js
module.exports = app => {
  const books = require("../controllers/book.controller");
  // cara mengakses gambar
http://polibatam.ac.id/img/perpustakaan.png
  // Mengambil semua data
  app.get("/api/books", books.findAll);
  // Mengambil data buku yang memiliki id = id
  app.get("/api/books/:id", books.findOne);
  // Membuat buku baru
  app.post("/api/books", books.create);
  // Mengubah data buku yang memiliki id = id
  app.put("/api/books/:id", books.update);
  // Hapus data buku yang memiliki id = id
  app.delete("/api/books/:id", books.delete);
  // Hapus seluruh data
  app.delete("/api/books", books.deleteAll);
};
```

7. Jika sudah lakukan perubahan, jangan lupa untuk push perubahan ke server

```
git add .
git commit -m "make it better"
git push heroku master
```

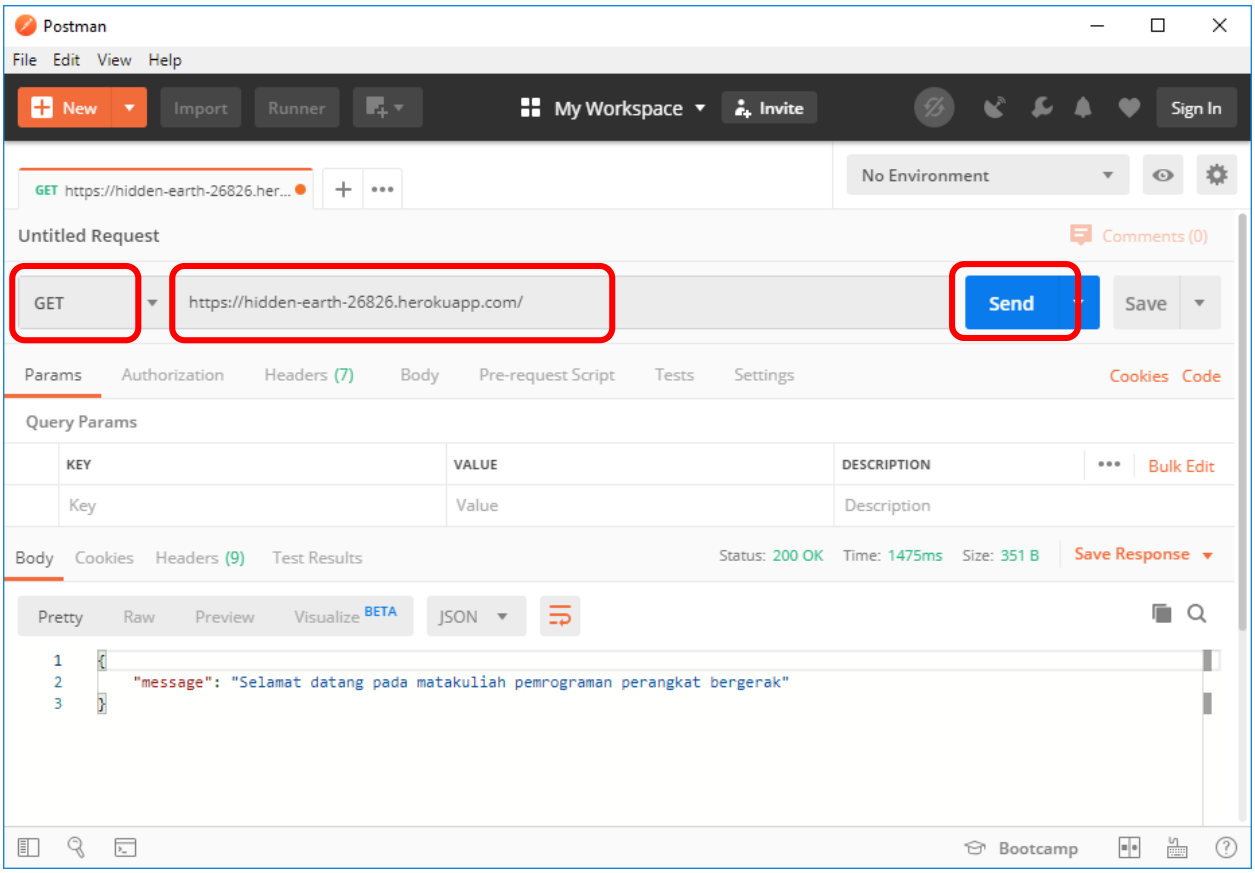
8. Buka aplikasi postman
9. Cara mudah melihat menjalankan aplikasi, jalankan perintah pada terminal:

```
heroku open
```



10. Buka aplikasi postman

11. Copy url lalu pilih method get, dan jalankan

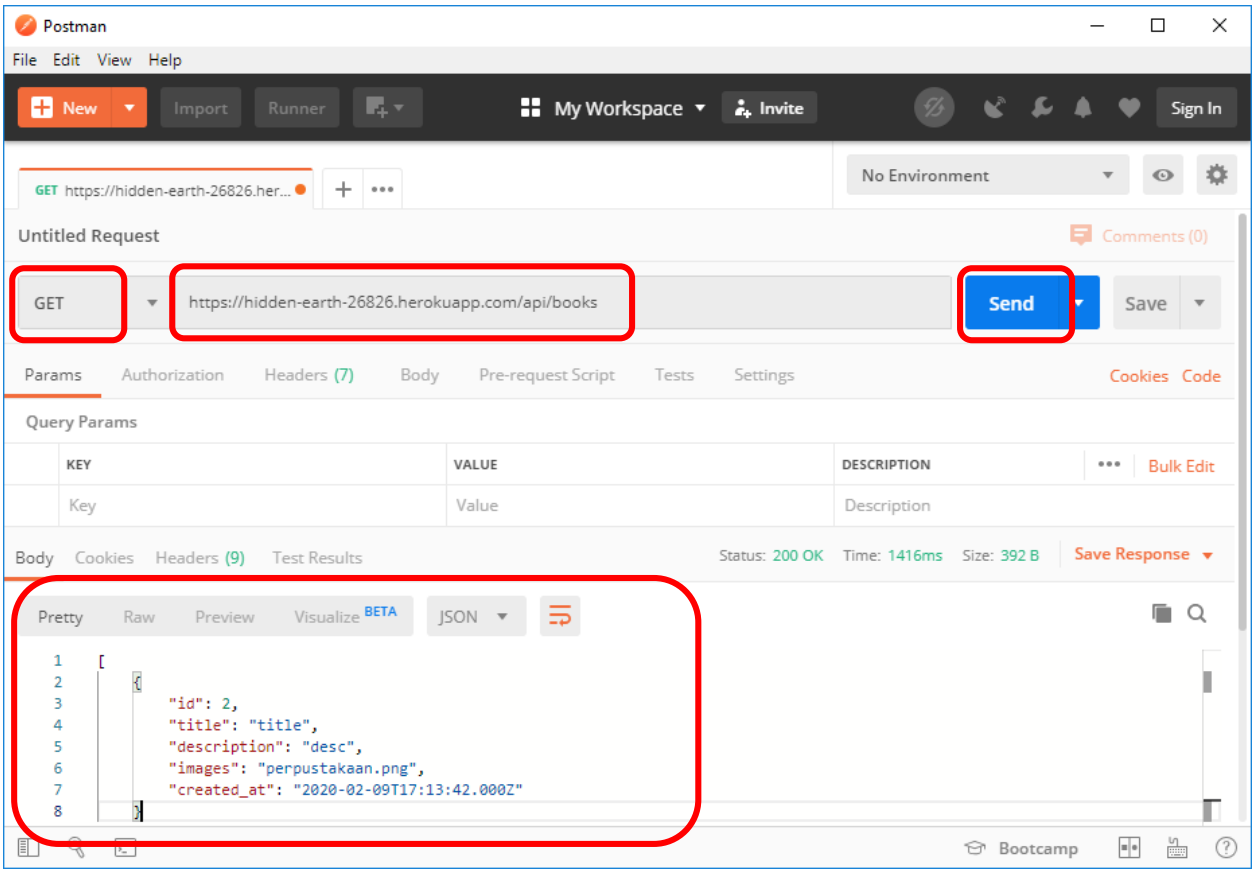


Contoh:

Method: GET

URL: https://hidden-earth-26826.herokuapp.com/api/books/2

12. Mengambil data buku

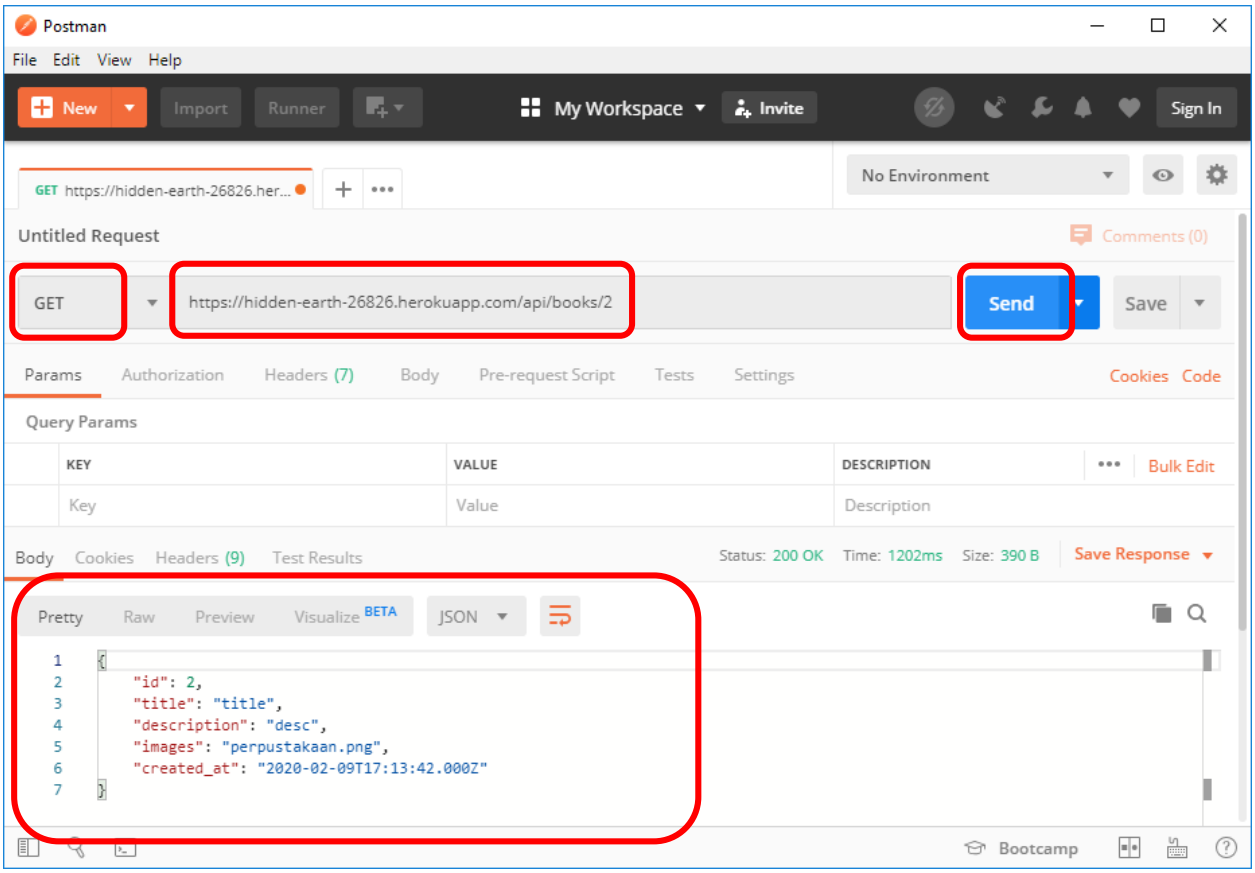


Contoh:

Method: GET

URL: https://hidden-earth-26826.herokuapp.com/api/books

13. Mengambil data buku berdasarkan id

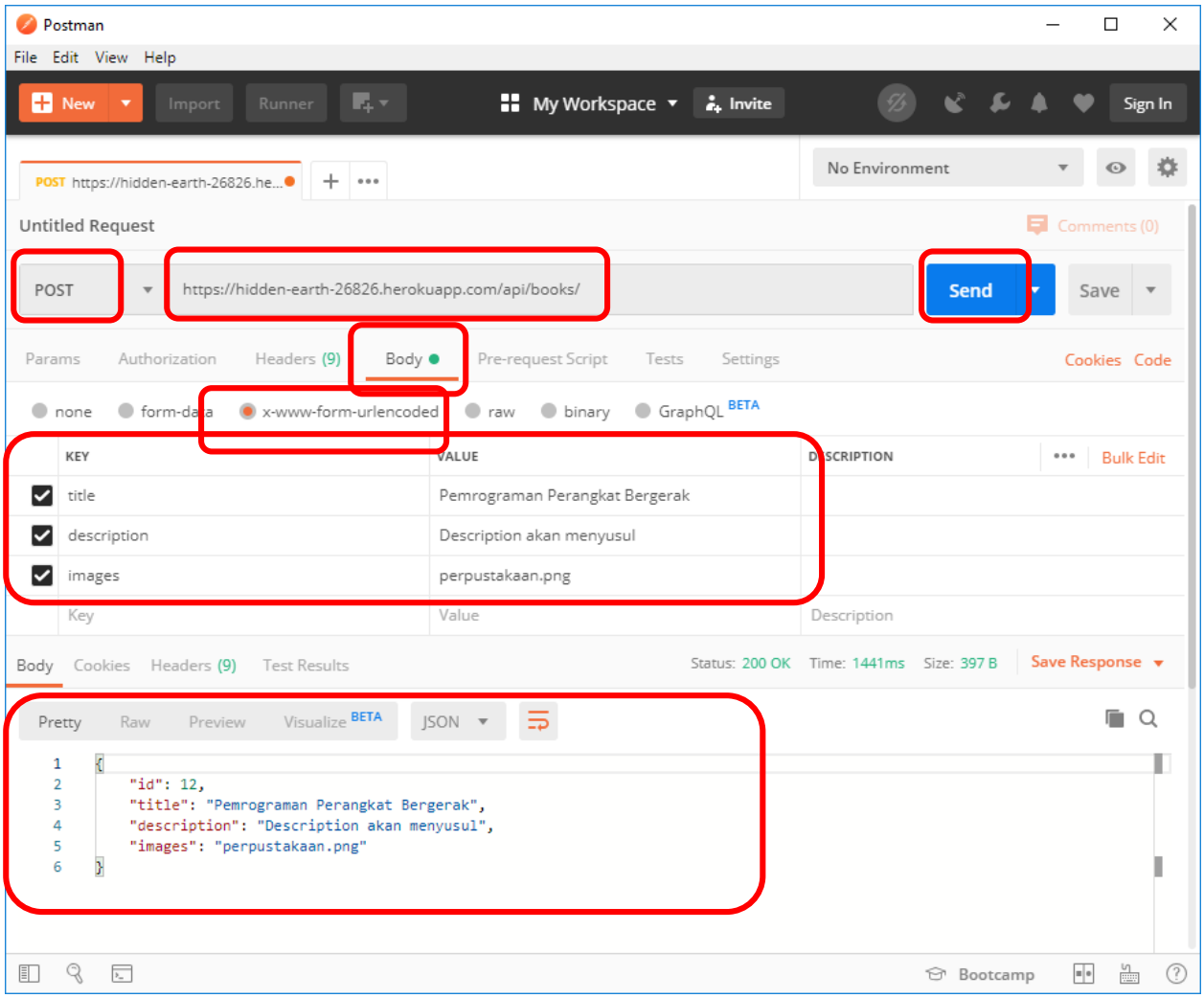


Contoh:

Method: GET

URL: https://hidden-earth-26826.herokuapp.com/api/books/2

14. Menambahkan data buku



Contoh:

Method: POST

URL: https://hidden-earth-26826.herokuapp.com/api/books/

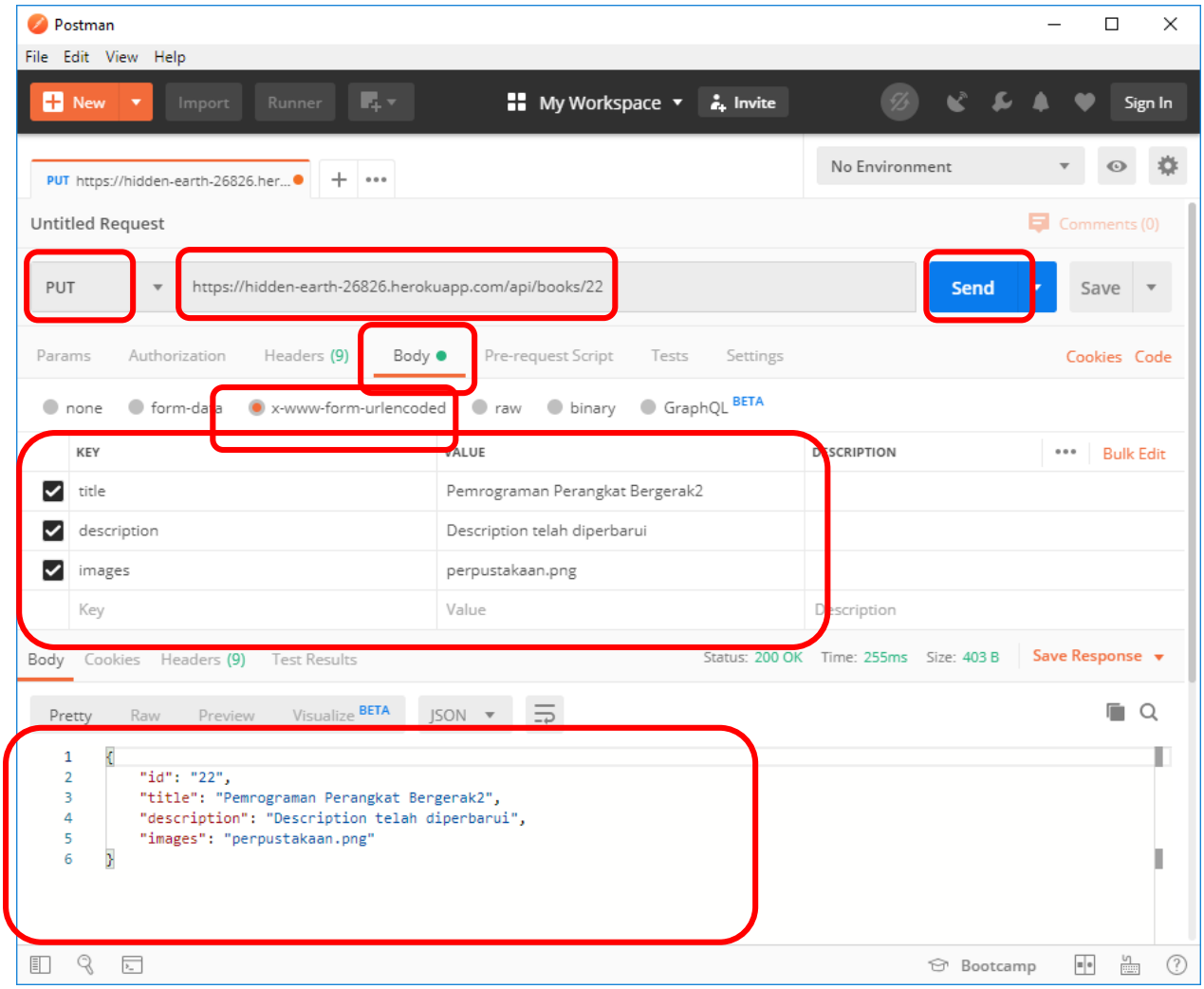
Body > x-www-form-urlencoded

title: Pemrograman Perangkat Bergerak

description: Description akan menyusul

images: perpustakaan.png

15. Memperbarui data



Contoh:

Method: PUT

URL: https://hidden-earth-26826.herokuapp.com/api/books/22

Body > x-www-form-urlencoded

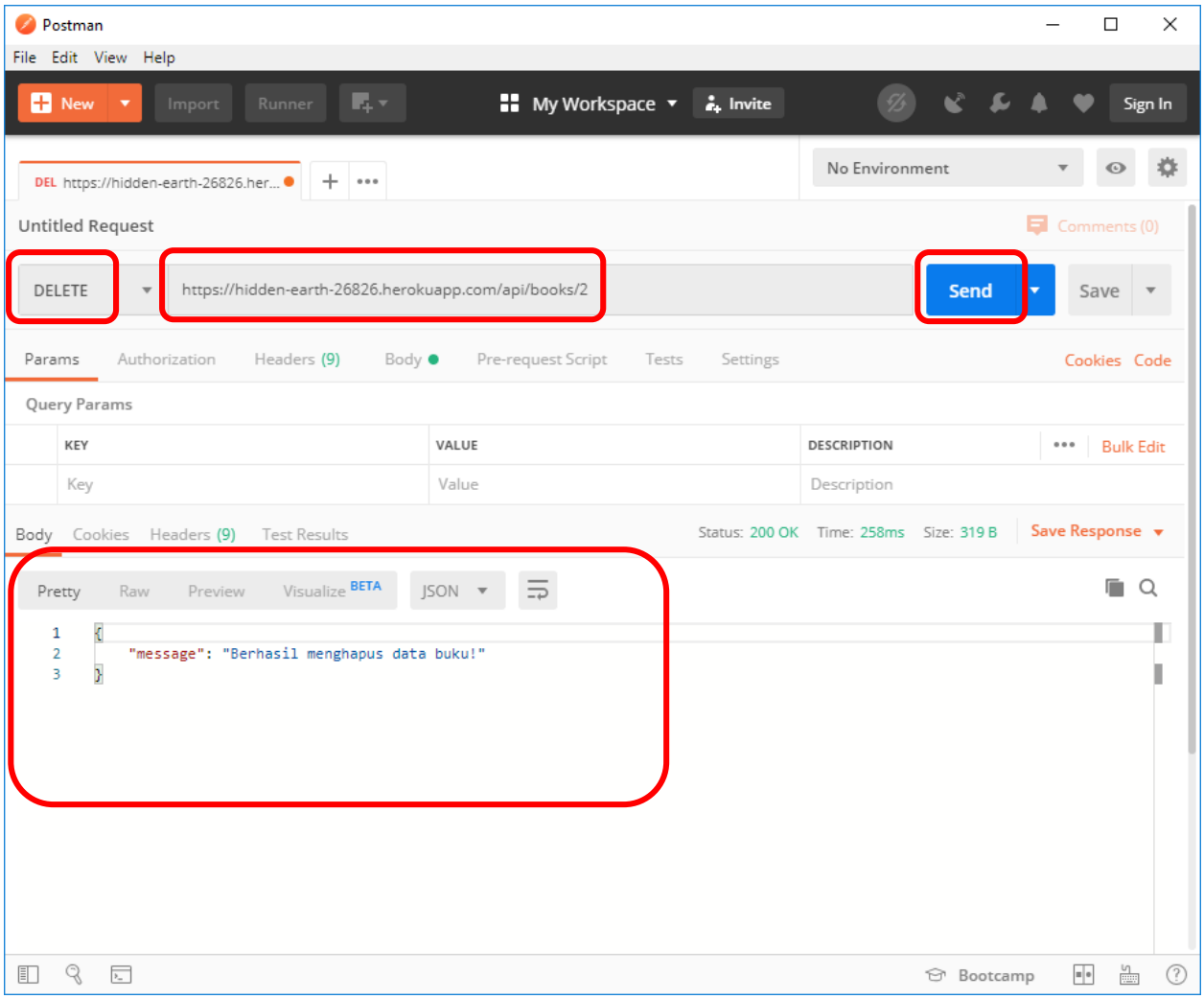
title: Pemrograman Perangkat Bergerak2

description: Description telah diperbarui

images: perpustakaan.png

id: 22

16. Menghapus data buku berdasarkan id

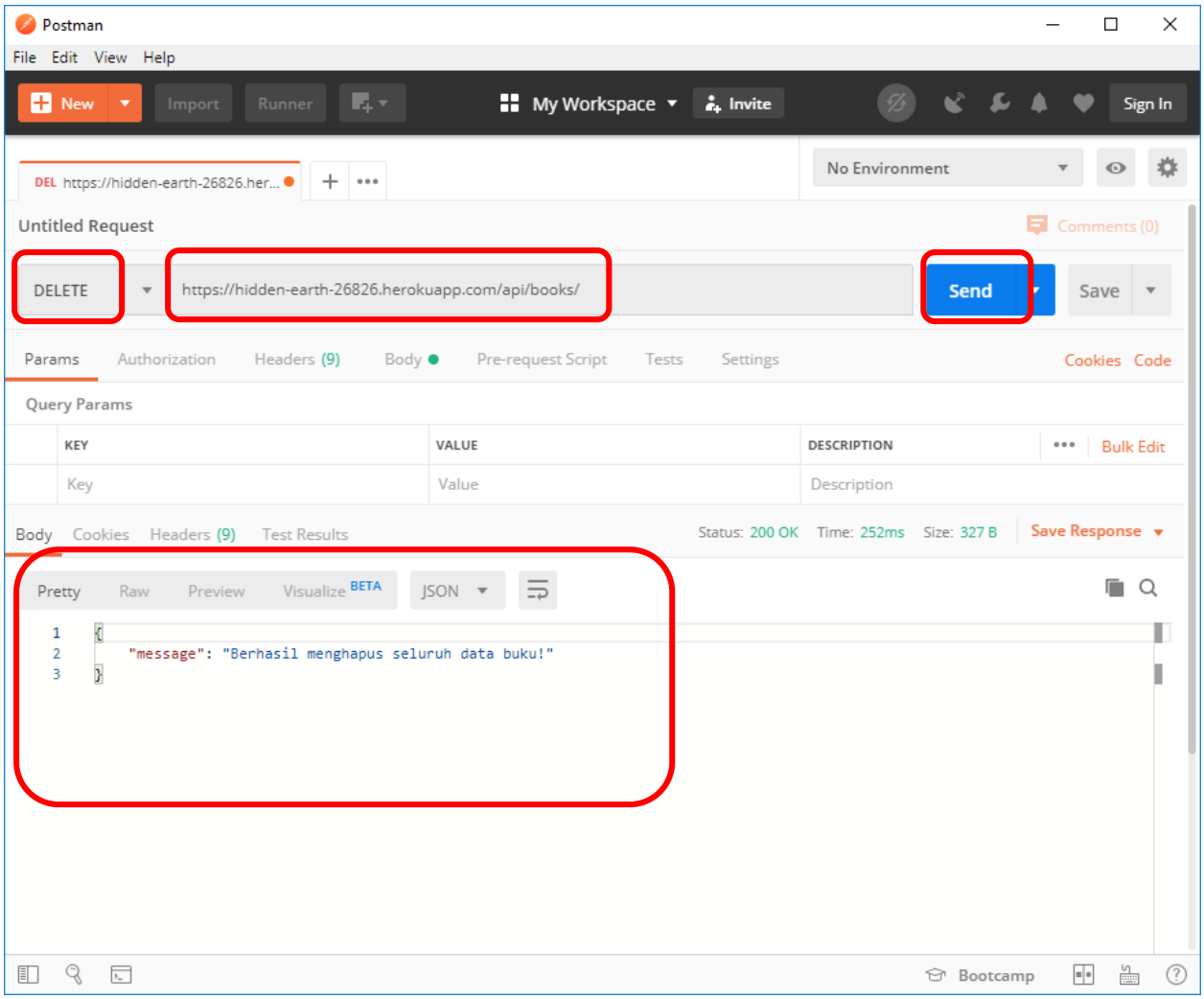


Contoh:

Method: DELETE

URL: https://hidden-earth-26826.herokuapp.com/api/books/2

17. Menghapus seluruh data buku



Contoh:

Method: DELETE

URL: https://hidden-earth-26826.herokuapp.com/api/books/

TUGAS

1. Selesaikan praktikum (point 30)
2. Implementasikan pada data yang berbeda, contoh peminjaman, user, author atau pembelian.
    - Menampilkan seluruh data (point 10)
    - Menampilkan data berdasarkan id (point 10)
    - Menambahkand data (point 10)
    - Mengupdate data berdasarkan id (point 10)
    - Menghapus data berdasarkan id (point 10)
    - Menghapus seluruh data (point 10)
3. Video (point 5) dan screenshot (point 5)

Setoran:

1. Folder utama beri nama > P03_NIM_NAMA
2. Didalam folder utama, terdapat beberapa folder yaitu
    - demo
        - gunakan screen recorder seperti obs (https://obsproject.com/), camtasia (https://www.techsmith.com/video-editor.html), bandicam (https://www.bandicam.com/)
        - format(.mp4)
    - project,
    - screenshot,

*Setoran diterima jika lengkap