Daniel Gorman

Service Layer

My Pet Pal

# Overview

The backend services for my web application will be built using the Express library from NodeJS. I have created a simple RESTful API which will be manipulated by the front end to store and change data.

The backend is broken up into three layers.

1. Route: This will define the paths that are available and can be used for specific information.

2. Controller: The controller will handle incoming HTTP requests and returns a response.

3. Service: The controllers will call the service methods that have logic to handle the data that is being requested or sent. Methods on the layer will be used to handle the database and manipulate the data.

Get usernames and get passwords:

```
//get all usernames

app.get("/usernames", async (req, res) => {
    try {
      const allUsernames = await pool.query("SELECT user_name FROM User");

      res.json(allUsernames.rows);
    } catch (err) {
      console.error(err.message);
    }
});


//get all paswords

app.get("/passwords", async (req, res) => {
  try {
    const allUsernames = await pool.query("SELECT user_password FROM User");

    res.json(allUsernames.rows);
  } catch (err) {
    console.error(err.message);
  }
});

//get all emails
```

The purpose is when an admin selects the table they can see all passwords and usernames for management of data if user's have trouble accessing their account.

Get pets:

```
//get all pets

app.get("/pets", async (req, res) => {
    try {
      const allPetnames = await pool.query("SELECT pet_name FROM PetAccount");

      res.json(allPetnames.rows);
    } catch (err) {
      console.error(err.message);
    }
});
```

The purpose is so that when a user logs in and goes to the home page they will be able to see a list of all their pets along with the information needed for their care.

Get Emails:

```
//get all emails

app.get("/emails", async (req, res) => {
    try {
      const allEmails = await pool.query("SELECT email FROM User");

      res.json(allEmails.rows);
    } catch (err) {
      console.error(err.message);
    }
});
```

The purpose is so an admin can get all emails in the case that a user has trouble logging in.

Get pet appointment and weight

```
//get pet weight

app.get("/petweight/:id", async (req, res) =>{
    const {id}= req.params
    try {
      const getPetweight = await pool.query("SELECT pet_weight FROM PetAccount WHERE pet_id = $1", [
          id
      ]);
    } catch (err) {
        console.error(err.message);
    }
});


//get pet appointment

app.get("/petppointment/:id", async (req, res) =>{
    const {id}= req.params
    try {
      const getAppointment = await pool.query("SELECT appointment_date FROM appointment WHERE pet_id = $1", [
          id
      ]);
    } catch (err) {
        console.error(err.message);
    }
});
```

Purpose: This is used once again to display in the home screen when a user logs in and they can see their pet's weight and appointments. This is useful for the diligent pet owner who needs to check their pet's weight progress.

Create a username (Post)

```
//create a username

app.post("/username", async(req, res) => {
    try {
      const { UserName } = req.body;
      const newUsername = await pool.query(
          "INSERT INTO User (user_name) VALUES ($1) RETURNING *",
          [UserName]
      );

      res.json(newUsername);
    } catch (err) {
      console.error(err.message);
    }
});
```

This allows a user to create a username when logging in for the first time or to give a secondary user a username.

Create a password (Post)

```
//create a password

app.post("/password", async(req, res) => {
  try {
    const { PassWord } = req.body;
    const newPassWord = await pool.query(
        "INSERT INTO User (pass_word) VALUES ($1) RETURNING *",
        [PassWord]
    );

    res.json(newPassWord);
  } catch (err) {
    console.error(err.message);
  }
});
```

The purpose is to allow the user to create a password for their account when logging in for the first time.

Create a pet account (Post)

```
//create a pet account

app.post("/petaccount", async(req, res) => {
    try {
      const { PetName } = req.body;
      const newPetaccount = await pool.query(
          "INSERT INTO PetAccount (pet_name) VALUES ($1) RETURNING *",
          [PetName]
      );

      res.json(newPetname);
    } catch (err) {
      console.error(err.message);
    }
});
```

The purpose is to allow the user to be able to create a pet profile for each of their pets. This would allow the pet's name to be logged and associated with the user's account

Create an email (Post)

```
//create an email

app.post("/email", async(req, res) => {
    try {
      const { Email } = req.body;
      const newEmail = await pool.query(
          "INSERT INTO User (email) VALUES ($1) RETURNING *",
          [Email]
      );

      res.json(newEmail);
    } catch (err) {
      console.error(err.message);
    }
});
```

The purpose of this is to allow the user to add an email to be associated with their account and kept track of in the database.

Create a pet appointment (Post)

```
//add pet appointment

app.post("/petappointment/:id", async(req, res) => {
    try {
        const { pet_id } = req.params; //WHERE
        const { pet_appointment } = req.body; //SET

        const updatePetWeight = await pool.query("UPDATE appointment SET appointment_date = $1 WHERE pet_id = $2", [pet_appointment, pet_id


        res.json("pets feeding time was added");
      } catch (err) {
        console.error(err.mesage);
      }
});
```

The purpose of this is to allow the user to add an appointment to their pet's profile or calendar on the main page, so that they can be reminded of the appointment via a email notification.

Update a username and password (Put)

```
//update a username

app.put("/usernames/:id", async (req, res) => {
    try {
      const { id } = req.params; //WHERE
      const { username } = req.body; //SET

      const updateUsername = await pool.query("UPDATE User SET user_name = $1 WHERE user_id = $2", [username, id]);


      res.json("username was updated");
    } catch (err) {
      console.error(err.mesage);
    }
});

//update a password

app.put("/password/:id", async (req, res) => {
  try {
    const { id } = req.params; //WHERE
    const { password } = req.body; //SET

    const updatePassword = await pool.query("UPDATE User SET user_password = $1 WHERE user_id = $2", [password, id]);


    res.json("password was updated");
  } catch (err) {
    console.error(err.mesage);
```

The purpose of this is so that the user can update their username and password if they wish. Either for security reasons, or in the event that a username or password was forgotten.

Update an email (Put)

```
//update an email

app.put("/email/:id", async (req, res) => {
    try {
      const { id } = req.params; //WHERE
      const { Email } = req.body; //SET

      const updateEmail = await pool.query("UPDATE User SET email = $1 WHERE user_id = $2", [Email, id]);


      res.json("users email was updated");
    } catch (err) {
      console.error(err.mesage);
    }
});
```

The purpose for this is to allow the user to update their email address if the need arises.

Update a pet weight (Put)

```
app.put("/petweight/:id", async (req, res) => {
    try {
      const { id } = req.params; //WHERE
      const { petweight } = req.body; //SET

      const updatePetweight = await pool.query("UPDATE PetAccount SET pet_weight = $1 WHERE pet_id = $2", [petweight, id]);


      res.json("Pet weight was updated");
    } catch (err) {
      console.error(err.mesage);
    }
});
```

The purpose of this is to allow a user to update their pet's weight from the edit pet tab on the website.

Update a pet appointment (Put)

```
//update a pet appointment

app.put("/appointment/:id", async (req, res) => {
    try {
      const { id } = req.params; //WHERE
      const { appointmentdate } = req.body; //SET

      const updateappointmentdate = await pool.query("UPDATE appointment SET appointment_date = $1 WHERE pet_id = $2", [appointmentdate, id

      res.json("appointment was updated");
    } catch (err) {
      console.error(err.mesage);
    }
});
```

The purpose of this is to update a pet appointment if the need arises.

Delete user and email (Delete)

```
//delete a username

app.delete("/user/:id", async (req, res) => {
    try {
      const { id } = req.params;
      const deleteUsername = await pool.query("DELETE FROM User WHERE user_id = $1", [id])
      res.json("Username was successfully deleted!");
    } catch (err) {
      console.error(err.message);
    }
});

//delete an email

app.delete("/email/:id", async (req, res) => {
    try {
      const { id } = req.params;
      const deleteEmail = await pool.query("DELETE FROM User WHERE user_id = $1", [id])
      res.json("Email was successfully deleted!");
    } catch (err) {
      console.error(err.message);
    }
});
```

The purpose of this is for the times when a user might want to delete an old email or delete a secondary user they have given certain permissions for.

Delete a pet and appointment (Delete)

```
//delete a pet

app.delete("/petaccount/:id", async (req, res) => {
    try {
      const { id } = req.params;
      const deletePet = await pool.query("DELETE FROM * WHERE pet_id = $1", [id])
      res.json("Pet was removed");
    } catch (err) {
      console.error(err.message);
    }
});
```

```
//delete appointment

app.delete("/appointment/:id", async (req, res) => {
    try {
      const { id } = req.params;
      const deleteAppointment = await pool.query("DELETE FROM appointment WHERE pet_id = $1", [id])
      res.json("Appointment was successfully deleted!");
    } catch (err) {
      console.error(err.message);
    }
});
```

For the unfortunate time may come when a pet has passed on or gone to a new home, this gives the user an option to delete that pet. Delete an appointment is used for the occasion that an appointment is canceled and no reminder is wanted.