```
'use strict';

const pg = require('pg');
const fs = require('fs');
const express = require('express');
const bodyParser = require('body-parser');
const PORT = process.env.PORT || 3000;
const app = express();

// NOTE: Your conString is dependent on your OS, will look something like ...
// const conString = 'postgres://USER:PASSWORD@HOST:PORT/DBNAME';

const client = new pg.Client(conString);

// REVIEW: Use the client object to connect to our DB.
client.connect();

// REVIEW: Install the middleware plugins so that our app can use the body-parser module.
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
app.use(express.static('./public'));

// REVIEW: Routes for requesting HTML resources
app.get('/new', (request, response) => {
  response.sendFile('new.html', {root: './public'});
});

// REVIEW: Routes for making API calls to use CRUD Operations on our database
app.get('/articles', (request, response) => {
  client.query('SELECT * FROM articles')
    .then(result => {
      response.send(result.rows);
    })
    .catch(err => {
      console.error(err)
    })
});

app.post('/articles', (request, response) => {
  client.query(
    `INSERT INTO
    articles(title, author, "authorUrl", category, "publishedOn", body)
    VALUES ($1, $2, $3, $4, $5, $6);
    `,
    [
      request.body.title,
      request.body.author,
      request.body.authorUrl,
      request.body.category,
      request.body.publishedOn,
      request.body.body
    ]
  )
    .then(() => {
      response.send('insert complete')
    })
    .catch(err => {
      console.error(err);
    });
});
```

```
app.put('/articles/:id', (request, response) => {
  client.query(
    `UPDATE articles
    SET
    title=$1, author=$2, "authorUrl"=$3, category=$4, "publishedOn"=$5, body=$6
    WHERE article_id=$7;
    `,
    [
      request.body.title,
      request.body.author,
      request.body.authorUrl,
      request.body.category,
      request.body.publishedOn,
      request.body.body,
      request.params.id
    ]
  )
    .then(() => {
      response.send('update complete')
    })
    .catch(err => {
      console.error(err);
    });
});

app.delete('/articles/:id', (request, response) => {
  client.query(
    `DELETE FROM articles WHERE article_id=$1;`,
    [request.params.id]
  )
    .then(() => {
      response.send('Delete complete')
    })
    .catch(err => {
      console.error(err);
    });
});

app.delete('/articles', (request, response) => {
  client.query(
    'DELETE FROM articles;'
  )
    .then(() => {
      response.send('Delete complete')
    })
    .catch(err => {
      console.error(err);
    });
});

loadDB();

app.listen(PORT, () => {
  console.log(`Server started on port ${PORT}!`);
});
```

```
//////// ** DATABASE LOADER ** ////////
////////////////////////////////////////
function loadArticles() {
  client.query('SELECT COUNT(*) FROM articles')
    .then(result => {
         if(!parseInt(result.rows[0].count)) {
        fs.readFile('./public/data/hackerIpsum.json', (err, fd) => {
          JSON.parse(fd.toString()).forEach(ele => {
            client.query(`
              INSERT INTO
              articles(title, author, "authorUrl", category, "publishedOn", body)
              VALUES ($1, $2, $3, $4, $5, $6);
              `,
              [ele.title, ele.author, ele.authorUrl, ele.category, ele.publishedOn,
ele.body]
            )
          })
        })
      }
    })
}

function loadDB() {
  client.query(`
    CREATE TABLE IF NOT EXISTS articles (
      article_id SERIAL PRIMARY KEY,
      title VARCHAR(255) NOT NULL,
      author VARCHAR(255) NOT NULL,
      "authorUrl" VARCHAR (255),
      category VARCHAR(20),
      "publishedOn" DATE,
      body TEXT NOT NULL);`
  )
    .then(() => {
      loadArticles();
    })
    .catch(err => {
      console.error(err);
    });
}
```