BizTime Solution

Springboard

Download solution

Core App Stuff

server.js

```
/** Server startup for BizTime. */


const app = require('./app');


app.listen(3000, function () {
  console.log("Listening on 3000");
});
```

app.js

```
/** BizTime express application. */


const express = require("express");

const ExpressError = require("./expressError")
const companiesRoutes = require("./routes/companies");
const invoicesRoutes = require("./routes/invoices");

const app = express();


app.use(express.json());
app.use("/companies", companiesRoutes);
app.use("/invoices", invoicesRoutes);


/** 404 handler */

app.use(function (req, res, next) {
  const err = new ExpressError("Not found", 404);
  return next(err);
});

/** general error handler */

app.use((err, req, res, next) => {
  res.status(err.status || 500);

  return res.json({
    error: err,
    message: err.message
  });
});


module.exports = app;
```

db.js

```
/** Database setup for BizTime. */


const { Client } = require("pg");

const client = new Client({
  connectionString: "postgresql:///biztime"
});

client.connect();


module.exports = client;
```

Routes

routes/companies.js

```
/** Routes for companies. */


const express = require("express");
const slugify = require("slugify");
const ExpressError = require("../expressError")
const db = require("../db");

let router = new express.Router();


/** GET / => list of companies.
 *
 *  => {companies: [{code, name, descrip}, {code, name, descrip}, ...]}
 *
 */

router.get("/", async function (req, res, next) {
  try {
    const result = await db.query(
          `SELECT code, name
           FROM companies
           ORDER BY name`
    );

    return res.json({"companies": result.rows});
  }

  catch (err) {
    return next(err);
  }
});


/** GET /[code] => detail on company
 *
 *  =>  {company: {code, name, descrip, invoices: [id, ...]}}
 *
 */

router.get("/:code", async function (req, res, next) {
  try {
    let code = req.params.code;

    const compResult = await db.query(
          `SELECT code, name, description
           FROM companies
           WHERE code = $1`,
        [code]
    );

    const invResult = await db.query(
          `SELECT id
           FROM invoices
           WHERE comp_code = $1`,
        [code]
    );

    if (compResult.rows.length === 0) {
      throw new ExpressError(`No such company: ${code}`, 404)
    }

    const company = compResult.rows[0];
    const invoices = invResult.rows;

    company.invoices = invoices.map(inv => inv.id);

    return res.json({"company": company});
  }

  catch (err) {
    return next(err);
  }
});


/** POST / => add new company
 *
 * {name, descrip}  =>  {company: {code, name, descrip}}
 *
 */

router.post("/", async function (req, res, next) {
  try {
    let {name, description} = req.body;
    let code = slugify(name, {lower: true});

    const result = await db.query(
          `INSERT INTO companies (code, name, description)
           VALUES ($1, $2, $3)
           RETURNING code, name, description`,
        [code, name, description]);

    return res.status(201).json({"company": result.rows[0]});
  }

  catch (err) {
    return next(err);
  }
});


/** PUT /[code] => update company
 *
 * {name, descrip}  =>  {company: {code, name, descrip}}
 *
 */

router.put("/:code", async function (req, res, next) {
  try {
    let {name, description} = req.body;
    let code = req.params.code;

    const result = await db.query(
          `UPDATE companies
           SET name=$1, description=$2
           WHERE code = $3
           RETURNING code, name, description`,
        [name, description, code]);

    if (result.rows.length === 0) {
      throw new ExpressError(`No such company: ${code}`, 404)
    } else {
      return res.json({"company": result.rows[0]});
    }
  }

  catch (err) {
    return next(err);
  }
});


/** DELETE /[code] => delete company
 *
 * => {status: "added"}
 *
 */

router.delete("/:code", async function (req, res, next) {
  try {
    let code = req.params.code;

    const result = await db.query(
          `DELETE FROM companies
           WHERE code=$1
           RETURNING code`,
        [code]);

    if (result.rows.length === 0) {
      throw new ExpressError(`No such company: ${code}`, 404)
    } else {
      return res.json({"status": "deleted"});
    }
  }

  catch (err) {
    return next(err);
  }
});


module.exports = router;
```

routes/invoices.js

```
/** Routes for invoices. */


const express = require("express");
const ExpressError = require("../expressError")
const db = require("../db");

let router = new express.Router();


/** GET / => list of invoices.
 *
 *  => {invoices: [{id, comp_code}, ...]}
 *
 */

router.get("/", async function (req, res, next) {
  try {
    const result = await db.query(
          `SELECT id, comp_code
           FROM invoices
           ORDER BY id`
    );

    return res.json({"invoices": result.rows});
  }

  catch (err) {
    return next(err);
  }
});


/** GET /[id] => detail on invoice
 *
 *  =>  {invoices: {id,
 *                  amt,
 *                  paid,
 *                  add_date,
 *                  paid_date,
 *                  company: {code, name, description}}}
 *
 */

router.get("/:id", async function (req, res, next) {
  try {
    let id = req.params.id;

    const result = await db.query(
          `SELECT i.id,
                  i.comp_code,
                  i.amt,
                  i.paid,
                  i.add_date,
                  i.paid_date,
                  c.name,
                  c.description
           FROM invoices AS i
             INNER JOIN companies AS c ON (i.comp_code = c.code)
           WHERE id = $1`,
        [id]);

    if (result.rows.length === 0) {
      throw new ExpressError(`No such invoice: ${id}`,404)
    }

    const data = result.rows[0];
    const invoice = {
      id: data.id,
      company: {
        code: data.comp_code,
        name: data.name,
        description: data.description,
      },
      amt: data.amt,
      paid: data.paid,
      add_date: data.add_date,
      paid_date: data.paid_date,
    }

    return res.json({"invoice": invoice});
  }

  catch (err) {
    return next(err);
  }
});


/** POST / => add new invoice
 *
 * {comp_code, amt}  =>  {id, comp_code, amt, paid, add_date, paid_date}
 *
 */

router.post("/", async function (req, res, next) {
  try {
    let {comp_code, amt} = req.body;

    const result = await db.query(
          `INSERT INTO invoices (comp_code, amt)
           VALUES ($1, $2)
           RETURNING id, comp_code, amt, paid, add_date, paid_date`,
        [comp_code, amt]);

    return res.json({"invoice": result.rows[0]});
  }

  catch (err) {
    return next(err);
  }
});


/** PUT /[code] => update invoice
 *
 * {amt, paid}  =>  {id, comp_code, amt, paid, add_date, paid_date}
 *
 * If paying unpaid invoice, set paid_date; if marking as unpaid, clear paid_date.
 */

router.put("/:id", async function (req, res, next) {
  try {
    let {amt, paid} = req.body;
    let id = req.params.id;
    let paidDate = null;

    const currResult = await db.query(
          `SELECT paid
           FROM invoices
           WHERE id = $1`,
        [id]);

    if (currResult.rows.length === 0) {
      throw new ExpressError(`No such invoice: ${id}`, 404);
    }

    const currPaidDate = currResult.rows[0].paid_date;

    if (!currPaidDate && paid) {
      paidDate = new Date();
    } else if (!paid) {
      paidDate = null
    } else {
      paidDate = currPaidDate;
    }

    const result = await db.query(
          `UPDATE invoices
           SET amt=$1, paid=$2, paid_date=$3
           WHERE id=$4
           RETURNING id, comp_code, amt, paid, add_date, paid_date`,
        [amt, paid, paidDate, id]);

    return res.json({"invoice": result.rows[0]});
  }

  catch (err) {
    return next(err);
  }
});


/** DELETE /[code] => delete invoice
 *
 * => {status: "deleted"}
 *
 */

router.delete("/:id", async function (req, res, next) {
  try {
    let id = req.params.id;

    const result = await db.query(
          `DELETE FROM invoices
           WHERE id = $1
           RETURNING id`,
        [id]);

    if (result.rows.length === 0) {
      throw new ExpressError(`No such invoice: ${id}`, 404);
    }

    return res.json({"status": "deleted"});
  }

  catch (err) {
    return next(err);
  }
});


module.exports = router;
```