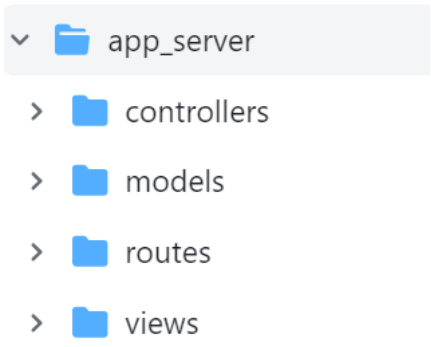
Examen-1 L3 Miage 23

- Soit le code nodeJS suivant

```
const express = require("express");
const app = express();
const port = 3000;

app.get("/", (req, res) =>
res.send("Hello Miage!"));

app.get("/products", (req,res) => {
  const products = [
    { id: 1,name: "hammer",},
    { id: 2,name: "screwdriver",},
    {id: 3,name: "wrench",},];
  res.json(products);
})

app.listen(port, () =>
console.log(`Example app listening
on port ${port}!`));
```

A) Transformez le code suivant pour utiliser la structuration MVC d'express.

🆘Correction



🔩model (products.js)

```
const products = [ { id: 1,name: "hammer",},
    { id: 2,name: "screwdriver",} …];

module.exports= { products}
```

Examen-1 L3 Miage 23

🔩App.js

```
const express = require("express");
const app = express();
const indexRouter = require("./app_server/routes/index");
app.use("/products", indexRouter);
```

🚀routes

```
const ctrlProducts = require('../controllers/products');
router.route('/')
    .get(ctrlProducts.productsReadAll)
```

🚀Controllers (products.js)

```
const { products } = require('../models/products');
const productsReadAll = (req, res) => {
    res.render('list', { products });
}
```

🔩views (list.pug)

```
  ol
    each product in products
      li
        a(href=`/products/${product.name}`)= product.name
```

Examen-1 L3 Miage 23

Code équivalent avec les cours (à la place de produits)

Voir le cours https://dupontexpressjs.blogspot.com/p/statique.html

code : https://github.com/dupontdenis/Cours-express-statique

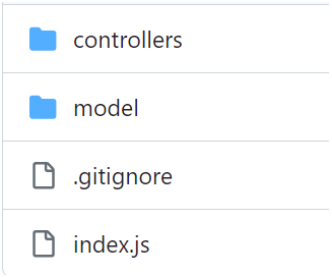
- Considérons la BD de l'exercice précédent définie par

```
const productsDB = {
  products: require("../model/products.json"),
  setProducts: function (data) {
    this.products = data;
  },};
```

B) Donnez le code de Création et de Mise à jour

```
const createNewProduct = async (body)
const updateProduct = async (body)
```

🆘Correction :



🔩Fichier model/product.js

```
[
    { id: 1,name: "hammer",}, ...
]
```

🔩fichier controllers//productsCtrl.js

Examen-1 L3 Miage 23

```
const productsDB = {
  products: require("../model/products.json"),
  setproducts: function (data) {
    this.products = data;
  },
};

const getAllproducts = () => {
  return productsDB.products;
};
```

```
1.  const createNewProduct = async (body) => {
2.    const newProduct = {
3.      id: productsDB.products?.length
4.        ? productsDB.products[productsDB.products.length - 1].id + 1
5.        : 1,
6.      name: body.name,
7.    };
8.
9.    productsDB.setproducts([...productsDB.products, newProduct]);
10.   //
11.   await fsPromises.writeFile(
12.     path.join(__dirname, "..", "model", "products.json"),
13.     JSON.stringify(productsDB.products)
14.   );
15.   return productsDB.products;
```

```
16.};
```

//🪛

```
1.  const updateProduct = async (body) => {
2.    const product = productsDB.products.find((p) => p.id ===
    parseInt(body.id));
3.    if (!product) {
4.      return { message: `Product ID ${body.id} not found` };
5.    }
6.    if (body.name) product.firstname = body.name;
7.     const filteredArray = productsDB.products.filter(
8.       (p) => p.id !== parseInt(body.id)
9.    );
10.  const unsortedArray = [...filteredArray, product];
11.  productsDB.setproducts(
12.    unsortedArray.sort((a, b) => (a.id > b.id ? 1 : a.id < b.id ? -1 : 0))
13.  );
14.  await fsPromises.writeFile(
15.    path.join(__dirname, "..", "model", "products.json"),
16.    JSON.stringify(productsDB.products)
17.  );
18.  return productsDB.products;
19.};
```

🚀fichier index.js

```
const fx = require("./controllers/productsCtrl");
```

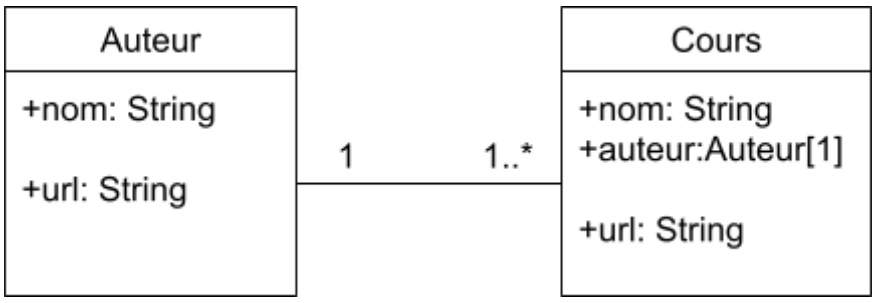
Examen-1 L3 Miage 23

```
const play = async () => {
  await fx.createNewProduct({ name: "adjustable wrench"});
  await fx.updateProduct({ id: 1, name: "screwdriver" });
};

play();
```

https://dupontl3.blogspot.com/2023/10/ds.html

https://github.com/dupontdenis/DS-JS-BD-REST/blob/master/controllers/usersController.js

- On s'intéresse aux modèles suivants :



C) Expliquez ce que l'on appelle des **Virtual properties !**

D) Comment gérer la relation entre les cours d'un même auteur ?

E) Quel est le rôle de la méthode "populate" ?

F) Donnez le code pour afficher pour chaque auteur la liste de ces cours.

"allCouresByAuthor,

🆘Correction :

Examen-1 L3 Miage 23

```
// Virtual for this book instance URL.
CourseSchema.virtual("url").get(function () {
  return "/book/" + this._id;
});
```

```
const AuthorSchema = new Schema({
  family_name: { type: String, required: true, maxLength: 100 },
});
const CourseSchema = new Schema({
  name: { type: String, required: true },
  author: { type: Schema.ObjectId, ref: "Author", required: true },
});
```

Rôle de populate

```
// Display list of all books.
exports.course_list = asyncHandler(async (req, res, next) => {
  const allCourses = await Course.find({}, "title author")
    .sort({ title: 1 })
    .populate("author")
    .exec();

  res.render("course_list", { title: "Course List", course_list: allCoursess });
});
```

Examen-1 L3 Miage 23

Ce qui permet d'avoir le nom des auteurs

```
ul
    each course in course_list
      li
        a(href=course.url) #{course.title}
         |   (#{course.author.name})
```

```
exports.author_detail = asyncHandler(async (req, res, next) => {
  // Get details of author and all their courses (in parallel)
  const [author, allCoursesByAuthor] = await Promise.all([
    Author.findById(req.params.id).exec(),
    Course.find({ author: req.params.id }, "name").exec(),
  ]);
  res.render("author_detail", {
    title: "Author Detail",
    author: author,
    author_courses: allCoursesByAuthor,
  });
});
```

Examen-1 L3 Miage 23

🪛Affichage

```
each course in author_courses
  dt
    a(href=course.url) #{course.name}
```