Linkera Not so Quick Audit

Routine audit linkera: 16 April 2024

[**Data Model 1**](#_ibfzt26w7ocv)

[**The backend 1**](#_ek4trf35llh0)

[1. Respect 1 post route 1 object created - high 2](#_xug9sodxfpvg)

[2. Joi in early phases medium 2](#_ypf4xjb0bla4)

[3. Save company data inside of annonce so that you don’t have to do too many things in the controllers medium 2](#_k73bl2jsrzkb)

[4. Get instead of post search on some routes medium 2](#_av4zcxxrbk3x)

[5. Small stupid Deb preference on not duplicating the name of the controller inside the routes. Just :id is enough in my opinion low 3](#_x4nazmoh7tkk)

[6. Mix of query / where ? very confusing and not useful medium 3](#_u5i5rq3fd8cw)

[7. File uploader not used high 3](#_tyqkwrx0pba3)

[8. Post object that returns a list high 4](#_kpdm8n4g19da)

[9. Language: why don't we use i18n? Question 4](#_ml608yi6xra4)

[10. Emails we’re not using brevo? I see an email folder? 5](#_dpgjg3shv138)

[11. Missing Credentials on local S 5](#_91sxdcwf9n7p)

[**Admin 5**](#_6gpjoagxod2)

[12. Missing Credentials on local S 5](#_usrcok24jxfr)

[15. Tailwind usage is not the best. 6](#_dlx83kd1gpxd)

[16. All of this logic could be inside of Infos with an early return. 6](#_vqybdmr7mrq9)

[17. Too many conditions in dom. Maybe it’s the time for componentizing some 6](#_m276x6hnvmw9)

[**The web app 6**](#_1gg1r2y6du9j)

[18. No separation of concerns. 7](#_dgpr93dy3wrg)

[19. Componentization of steps not good HIGH 7](#_q4mry9z4ox40)

[**The UX Feedbacks 9**](#_5f2303ocuw79)

# Data Model

Models le stud style ✅

# The backend

<https://github.com/selego/linkera/blob/main/api/src/controllers/annonce.js>

### Respect 1 post route 1 object created - high

I find this helps separate concerns.   
<https://github.com/selego/linkera/blob/main/api/src/controllers/annonce.js>  
The post above creates 3 objects. Is this necessary?

### Joi in early phases medium

Note: Copy paste from ithake

Using Joi in the early phases of a project, where frequent changes are common, is not optimal due to the added complexity and increased development time it introduces. The structured nature of Joi's validation hinders rapid iterations based on evolving requirements.

My real thoughts (Deb) (on ithake)  
  
Was I ok with using it? Yes  
Do I think we should have used it at this phase? No  
Did it slow us down? Yes  
Do I think it’s necessary at a more mature phase of the project? Yes  
  
It unfortunately makes the code look too messy at the moment and reduces code readability.

### 

### Save company data inside of annonce so that you don’t have to do too many things in the controllers medium

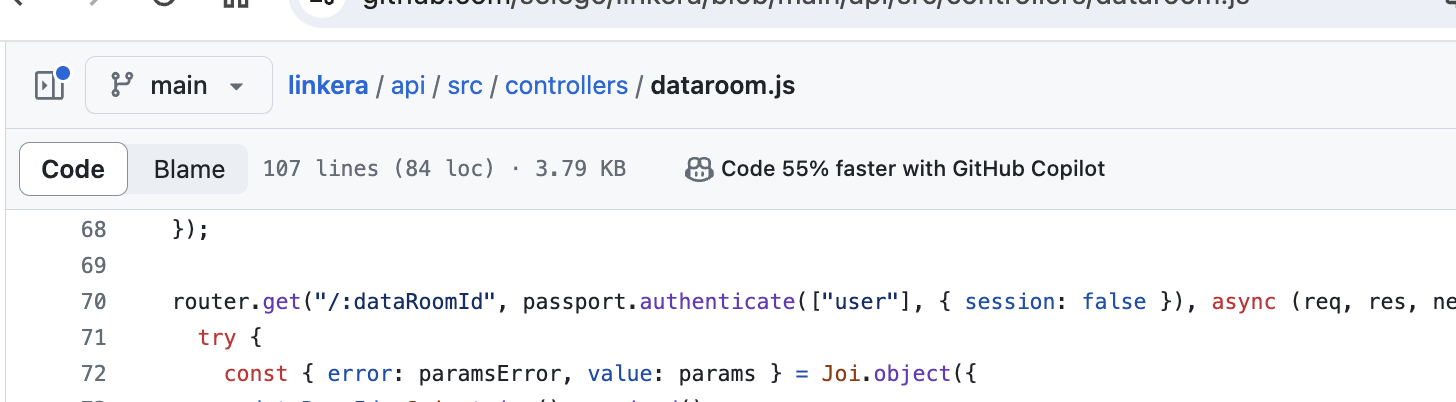
<https://github.com/selego/linkera/blob/main/api/src/controllers/annonce.js#L160>

( I think this will solve all of the issues above)

### Get instead of post search on some routes medium

I assume this is technical debt and you’re on it when you have time.   
<https://github.com/selego/linkera/blob/main/api/src/controllers/dataroom.js>

### Small stupid Deb preference on not duplicating the name of the controller inside the routes. Just :id is enough in my opinion low



### Mix of query / where ? very confusing and not useful medium

<https://github.com/selego/linkera/blob/main/api/src/controllers/document.js>

| **const** where = { deleted\_at: null };  **if** (query.data\_room\_id) where.data\_room\_id = query.data\_room\_id;  **else** **if** (user.role === "user") where.user\_id = user.\_id.toString();   **const** data = **await** DocumentModel.find(where).limit(query.size).skip(query.skip).sort({ created\_at: -1 });   **return** res.status(200).send({ ok: true, data });  } **catch** (error) {  next(error);  } }); |
| --- |

### File uploader not used high

const multer = require("multer");

const upload = multer();

router.post("/", passport.authenticate(["user"], { session: false }), upload.array("files"), **async** (req, res, next) => {  
 **try** {  
 **const** { error: queryError, value: query } = Joi.object({  
 data\_room\_id: Joi.string().allow("").optional(),  
 })  
 .unknown()  
 .validate(req.query);

Use the file controller and just save the url in this call.

### Post object that returns a list high

A post object should just return the object itself, not the list, i wouldn’t create many objects in the post route either ( which is happening here actually )

| router.post("/", passport.authenticate(["user"], { session: false }), upload.array("files"), **async** (req, res, next) => {  …   console.log("dataRoom", query.data\_room\_id, dataRoom);   **…**   console.log("documents", documents);  **await** DocumentModel.insertMany(documents);   **const** where = { deleted\_at: null };  **if** (dataRoom) where.data\_room\_id = dataRoom.\_id.toString();  **const** data = **await** DocumentModel.find(where).sort({ created\_at: -1 });  **return** res.status(200).send({ ok: true, data });  } **catch** (error) {  next(error);  } }); |
| --- |

### Language: why don't we use i18n? Question

### Emails we’re not using brevo? I see an email folder?

### Missing Credentials on local S

Increases time of onboarding

# Admin

### Missing Credentials on local S

Increases time of onboarding

1. Not sure config should be inside of services,confusing

Archi is nice overall btw

1. Fix smaller issues with the fetches

Naming, mix of french and english in the variables (enums), const {data,ok} = await is the pattern we usually use, then early return

| useEffect(() => {  **const** fetcgData = **async** () => {  setLoading(true);  **try** {  **const** query = **new** URLSearchParams();  query.append("type", "levee");  **if** (filters.status) query.append("status", filters.status);  **if** (filters.page) query.append("skip", (filters.page - 1) \* filters.pageSize);  **const** res = **await** api.post(`/annonce/search?${query.toString()}`);  **if** (res.ok) {  setAnnonces(res.data);  }  setLoading(false);  } **catch** (err) {  console.error(err);  }  };  fetcgData();  }, [filters]); |
| --- |

### Tailwind usage is not the best.

Do we really need to be as precise? Can't we just use a class they have instead of doing the whole absolute values.

<div className="w-[15%]">{new Date(e.created\_at).toLocaleDateString("fr-FR")}</div>

### All of this logic could be inside of Infos with an early return.

| {view === "infos" ? (  <Infos discussion={discussion} setDiscussion={setDiscussion} userTo={userTo} userFrom={userFrom} />  ) : (  <div className="col-span-2 bg-gray-900 rounded-lg p-5 flex flex-col gap-2">  <pre className="text-white text-sm">{JSON.stringify({ ...discussion }, null, 2)}</pre>  </div>  )} |
| --- |

### Too many conditions in dom. Maybe it’s the time for componentizing some

<https://github.com/selego/linkera/blob/main/admin/src/pages/discussion/View.jsx>

# The web app

Note: Same feedback as in admin applies.

### No separation of concerns.

If you’re fetching an annonce, fetch an annonce, why do you fetch the company at the same time? It complexifies the controller, complexifies the call, it introduces inconsistency inside of the data object that we send back from the controller.  
Extra points: async inside effect, no early return no {data,ok} destructuring

| useEffect(() => {  **const** fetchData = **async** () => {  **try** {  **const** res = **await** api.get(`/annonce/${annonceId}`);  **if** (res.ok) {  setAnnonce(res.data.annonce);  setCompany(res.data.company);  } **else** {  toast.error("Une erreur est survenue");  }  } **catch** (e) {  console.error(e);  }  };  fetchData();  }, [annonceId]); |
| --- |

Res1, res2 should be named better

| **const** handleSave = **async** () => {  **try** {  **const** res1 = **await** api.put(`/annonce/${annonceId}`, annonce);  **if** (res1.ok) {  **const** res2 = **await** api.put(`/company/${company.id}`, company); *// doesn't work on annonce creation, saves anyway somehow*  navigate("/home");  toast.success("Votre annonce a bien été sauvegardée");  }  } **catch** (error) {  console.error(error);  toast.error("Une erreur est survenue");  }  }; |
| --- |

### Componentization of steps not good HIGH

The mess is serious downstairs, not sure if it’s a wise idea to put the components into an array like this, why not make a component ?

| **const** STEPS = [  <CompanyDetails values={company} setValues={setCompany} setStep={setStep} />,  <Documents values={company} setValues={setCompany} setStep={setStep} onSave={handleSave} />,  <CompanyLocation values={company} setValues={setCompany} setStep={setStep} onSave={handleSave} />,  <FinancialData values={company} setValues={setCompany} setStep={setStep} onSave={handleSave} />,  annonce.type === "levee" ? (  <Funding values={annonce} setValues={setAnnonce} setStep={setStep} onSave={handleSave} />  ) : (  <SellingPrice values={annonce} setValues={setAnnonce} setStep={setStep} onSave={handleSave} />  ),  <Annonce values={annonce} setValues={setAnnonce} setStep={setStep} onSave={handleSave} onSubmit={handleSubmit} />,  ];   const STEPS\_LABEL = [  "Informations Générales",  "Documents",  "Localisation",  "Informations financières",  annonce.type === "levee" ? "Estimation" : "Prix de cession",  "Votre annonce",  ];   const STEPS\_ICON = [  <Bs1CircleFill className="mr-3" />,  <Bs2CircleFill className="mr-3" />,  <Bs3CircleFill className="mr-3" />,  <Bs4CircleFill className="mr-3" />,  <Bs5CircleFill className="mr-3" />,  <Bs6CircleFill className="mr-3" />,  ]; |
| --- |

# 

# The UX Feedbacks

No big issues on the tech-ux side.

# After Audit Annex

# API

1. to check whether we should do it in all our post search controllers or do something more secure  
   <https://github.com/selego/linkera/blob/main/api/src/controllers/company.js>

| Object.entries(body)  .filter(([key, value]) => key !== "limit" && key !== "skip" && value)  .forEach(([key, value]) => (where[key] = value)); |
| --- |