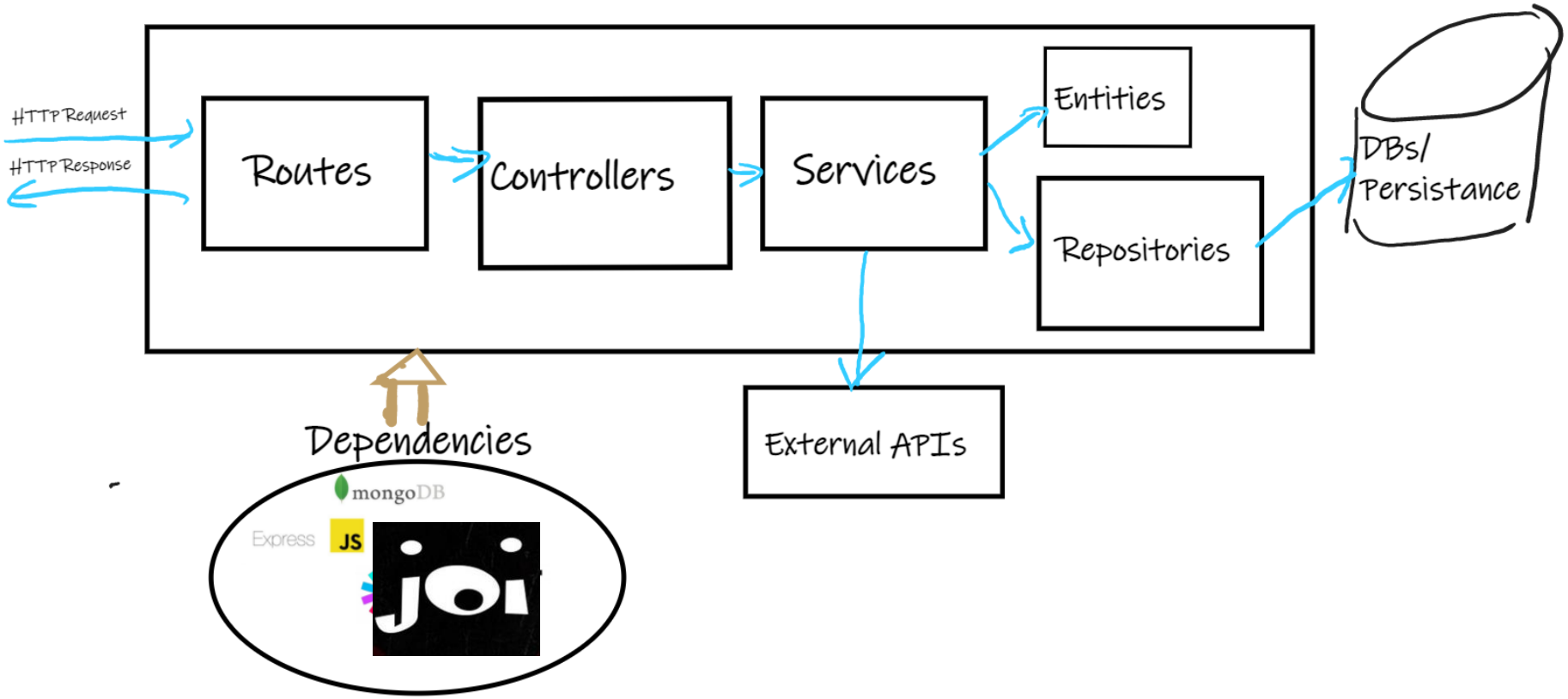
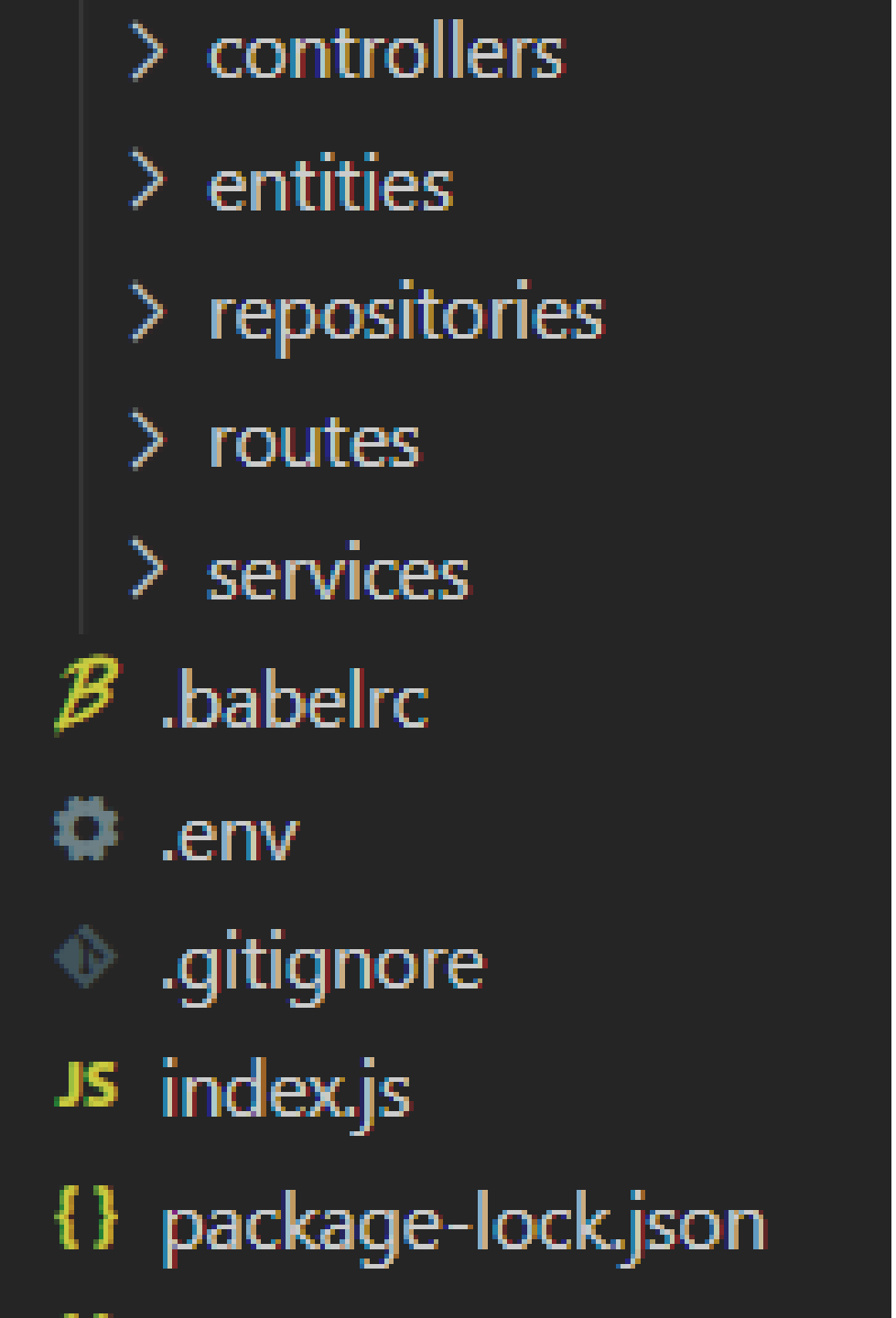
The Joi of Validation

And some other stuff

But first: Recap on Clean Architecture

Architecture we're following in the lab…





Last Weeks Example/Demo

Extend last weeks example to use in-memory DB

- Add User Entity
- Add User Repository
- Implement as In-Memory DB
- Add User Controller
- Add User Service

See https://github.com/fxwalsh/ewd-week9-example

Validation with Joi

Example User Entity

```
export default class User {
    constructor(id, name,  email,password, dob, type ) {
      this.id = id;
      this.name = name;
      this.email = email;
      this.password = password;
      this.dob = dob;
      this. type = type;
    }
  }
```

Add a few extra properties in addition to last week

Context

- When creating a User you don’t want this…

```
{
    "userName": "a",
    "name": "1234",
    "password": "password",
    "type": "A Friend",
    "dob": "12 Jul",
    "phone": "don't have one",
    "email": "jonndjg"
}
```

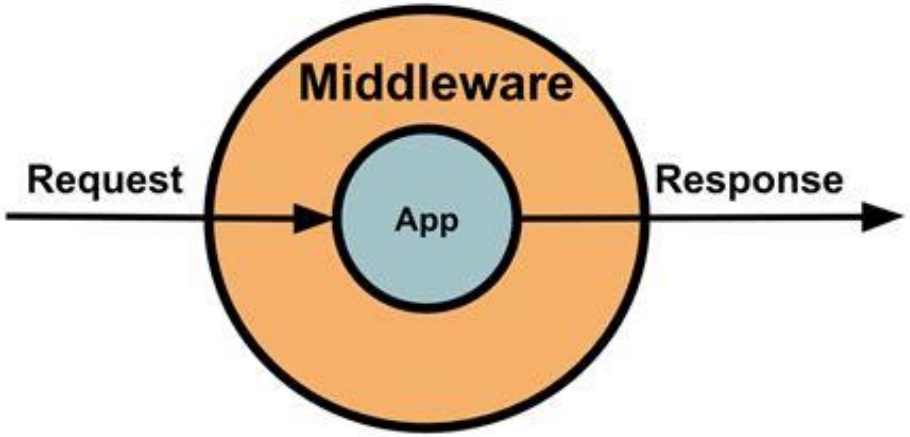
Bad Data:

- Name should be alphanumeric at least
- Password is weak
- Type should be FRIEND,FAMILY, or OTHER
- DOB has no year
- Phone is not a phone number
- Email is not formatted properly

Prevent bad data making its way into your app: Use Data Validation

Aside: Object Relational Mapping

- Mongoose, Sequelize, Knex provide for specification of validation constraints:
  - Handled in the application before persisting to DB
  - Good for enforcing Data layer rules (primary keys, no duplicate values)
  - Tightly coupled to the ORM implementation (what if I want to change DB?)
- In an API, data arriving via HTTP to endpoints
  - Good idea Validate at request level, using validation middleware/controller
  - In Express we can use **middleware** on the route.



Joi

- Use this module to validate data at the request level.

joi TS

17.6.0 • Public • Published 2 months ago

Readme | Explore BETA | 5 Dependencies | 8,610 Dependents | 210 Versions

joi

The most powerful schema description language and data validator for JavaScript.

Installation

```
npm install joi
```

Visit the joi.dev Developer Portal for tutorials, documentation, and support

Useful resources

- Documentation and API
- Versions status

Install

```
> npm i joi
```

Repository

github.com/sideway/joi

Homepage

github.com/sideway/joi#readme

Weekly Downloads

5,985,882

Using Joi: Define Schema

```
import Joi from 'joi';

const userSchema = Joi.object({
    email:
Joi.string().email().lowercase().required(),
    password: Joi.string().min(5).required(),
    name: Joi.string().min(1).required(),
    dob: Joi.string(),
    type: Joi.string()
});

export default  userSchema;
```

email is a String, formatted as email, lowercase, and must be present(required)

Password is a String, length of 4 or longer, and must be present

name is a String at least one character long

Note: This is not a good schema! We will improve later…

Using Joi: Include in Dependencies

- Include Schema as dependencies passed to Route

```
import InMemoryRepository from '../repositories/InMemoryRepository';
import userSchema from '../validators/userSchema';

const buildDependencies = () => {
  const dependencies = {
  };

  dependencies.userSchema = userSchema;

  if (process.env.DATABASE_DIALECT === "in-memory") {
    dependencies.usersRepository = new InMemoryRepository();
```

Extract from dependencies.js

In keeping with clean architecture approach, Validator schema introduced using dependency injection.

Validation Schema, Router

Validation Controller introduced onto route

Request Response cycle stops if Validation fails .

```javascript
export default (dependencies) => {

    const { userSchema } = dependencies;

    const validateUser = async (request, response, next) => {

        try {
            const validated = await userSchema.validateAsync(request.body);
            request.body = validated;
            next();
        } catch (err) {

            next(new Error(`Invalid Data ${err.message}`));
        }
    };

    return {
        validateUser
    };
};
```

userValidationController.js

```javascript
import express from 'express';
import userController from '../controllers/userController';
//Import Validation Controller
import userValidationController from '../controllers/userValidationController';

const createRouter = (dependencies) => {
    const router = express.Router();
    // load controller with dependencies
    const controller = userController(dependencies);
    const validationController = userValidationController(dependencies);

    //Add Controller(s) to routes
    router.route('/')
        .post(validationController.validateUser,controller.createUser);
```

userRouterController.js

Joi Validation: Dates

- Do ‘npm install @joi/date’ to install date format extension

```
import Joi from 'joi';
import JoiDate from '@joi/date';

const NewJoi = Joi.extend(JoiDate);

const userSchema = Joi.object({
    email: Joi.string().email().lowercase().required(),
    password: Joi.string().required().regex(/^(?=.*\d)(?=.*[!@#$%^&*
    name: Joi.string().min(1).required(),
    dob: NewJoi.date().format("DD/MM/YYYY"),
    type: Joi.string().valid('ADMIN','MEMBER')
});


export default  userSchema;
```

Adds Date extension to Joi

Checks date conforms to DD/MM/YYYY format

Joi Validation: Enumerations

- Specify list of valid values:

```
import Joi from 'joi';
import JoiDate from '@joi/date';

const NewJoi = Joi.extend(JoiDate);

const userSchema = Joi.object({
    email: Joi.string().email().lowercase().required(),
    password: Joi.string().required().regex(/^(?=.*\d)(?=.*[!@#$%^&*
    name: Joi.string().min(1).required(),
    dob: NewJoi.date().format("DD/MM/YYYY"),
    type: Joi.string().valid('ADMIN','MEMBER')
});


export default  userSchema;
```

Type must be in list of values

Joi Validation: Regular Expressions

- Use Regular expressions to validate password/phone properties

```
password:Joi.string().min(7).required().regex(/^(?=.*[A-Za-z])(?=.*\d)(?=.*[@$!%*#?&])[A-Za-z\d@$!%*#?&]{7,}$/),
phone:Joi.string().regex(/^\s*\+?\s*([0-9][\s-]*){9,}$/),
```

Phone number has length >9 and only contains numbers, spaces and '-'. Can begin with '+'

Ensures Password has length >7 and had letter, number and special char