SEBA Master: Web Application Engineering

Introduction to backend node-js techniques

Klym Shumaiev, 09.05.2016, MI HS 2

Software Engineering for Business Information Systems (sebis)
Department of Informatics
Technische Universität München, Germany

wwwmatthes.in.tum.de

Tutor

Felix Lachenmaier

- Email: felix.lachenmaier@tum.de
- Office hours: Every Thursday 8.00 – 15.45, Room MI 01.12.034
- Support for all technical issues

Outline

Version control

1. Always use version control
2. Always use version control
3. Always use version control

Version control

Have you ever:

Made a change to code, realized it was a mistake and wanted to revert back?

Lost code or had a backup that was too old?

Had to maintain multiple versions of a product?

Wanted to see the difference between two (or more) versions of your code?

Wanted to prove that a particular change broke or fixed a piece of code?

Wanted to review the history of some code?

Wanted to submit a change to someone else's code?

Wanted to share your code, or let other people work on your code?

Wanted to see how much work is being done, and where, when and by whom?

Wanted to experiment with a new feature without interfering with working code?

Version control

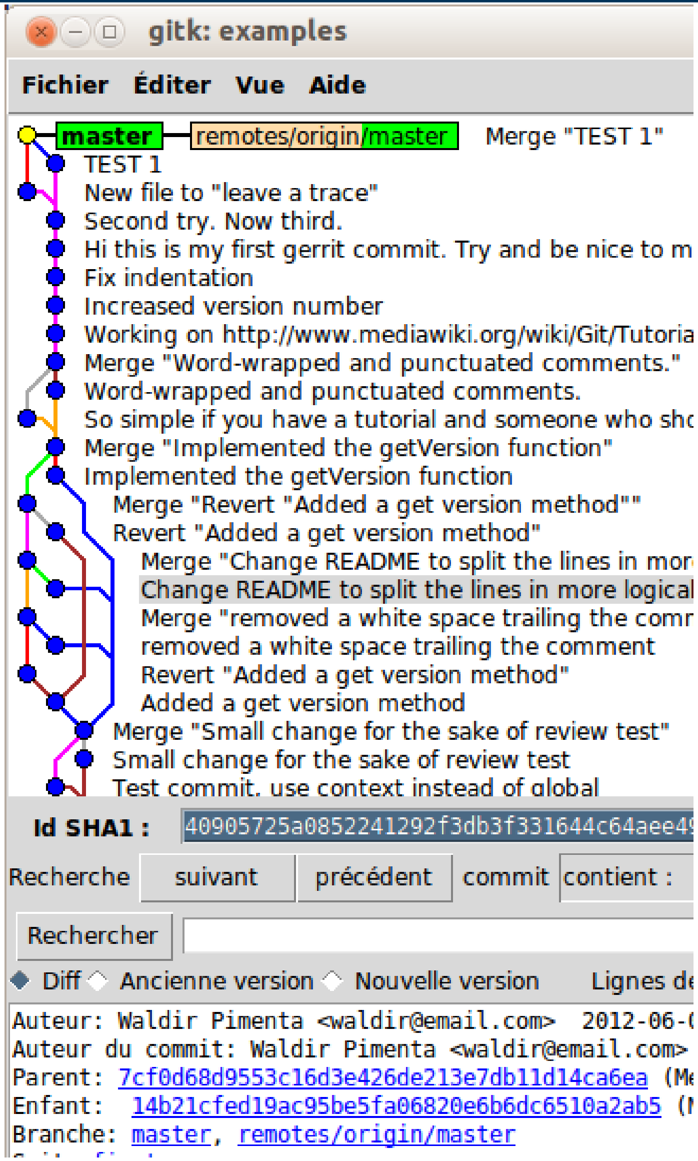
Branching

→ (Slightly) different versions of one project

→ Makes a lot of things easier

→ You should create new branches for each user and each feature

→ Use as few as possible, but as many as necessary



gitk: examples
Fichier Éditer Vue Aide
master
remotes/origin/master
Recherche suivant précédent commit contient :
Recherchер
Diff Ancienne version Nouvelle version Lignes de

```
Merge "TEST 1"
TEST 1
New file to "leave a trace"
Second try. Now third.
Hi this is my first gerrit commit. Try and be nice to m
Fix indentation
Increased version number
Working on http://www.mediawiki.org/wiki/Git/Tutoria
Merge "Word-wrapped and punctuated comments."
Word-wrapped and punctuated comments.
So simple if you have a tutorial and someone who sho
Merge "Implemented the getVersion function"
Implemented the getVersion function
Merge "Revert "Added a get version method""
Revert "Added a get version method"
Merge "Change README to split the lines in mor
Change README to split the lines in more logical
Merge "removed a white space trailing the comr
removed a white space trailing the comment
Revert "Added a get version method"
Added a get version method
Merge "Small change for the sake of review test"
Small change for the sake of review test
Test commit. use context instead of global
```

Id SHA1 : 40905725a0852241292f3db3f331644c64aee4

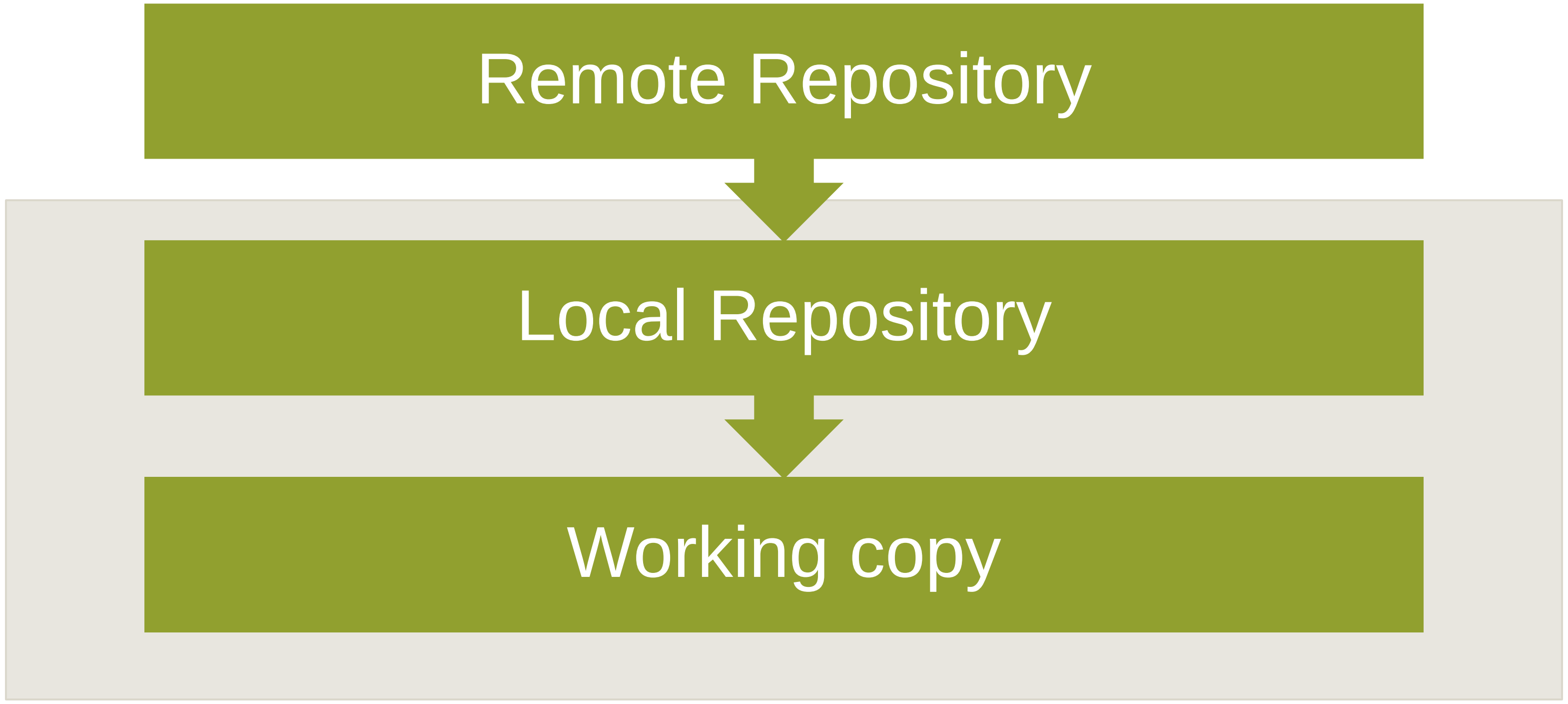
```
Auteur: Waldir Pimenta <waldir@email.com>  2012-06-(
Auteur du commit: Waldir Pimenta <waldir@email.com>
Parent: 7cf0d68d9553c16d3e426de213e7db11d14ca6ea (Me
Enfant:  14b21cfed19ac95be5fa06820e6b6dc6510a2ab5 (I
Branche: master, remotes/origin/master
```

Version control

sebis

Remote and Local Repositories

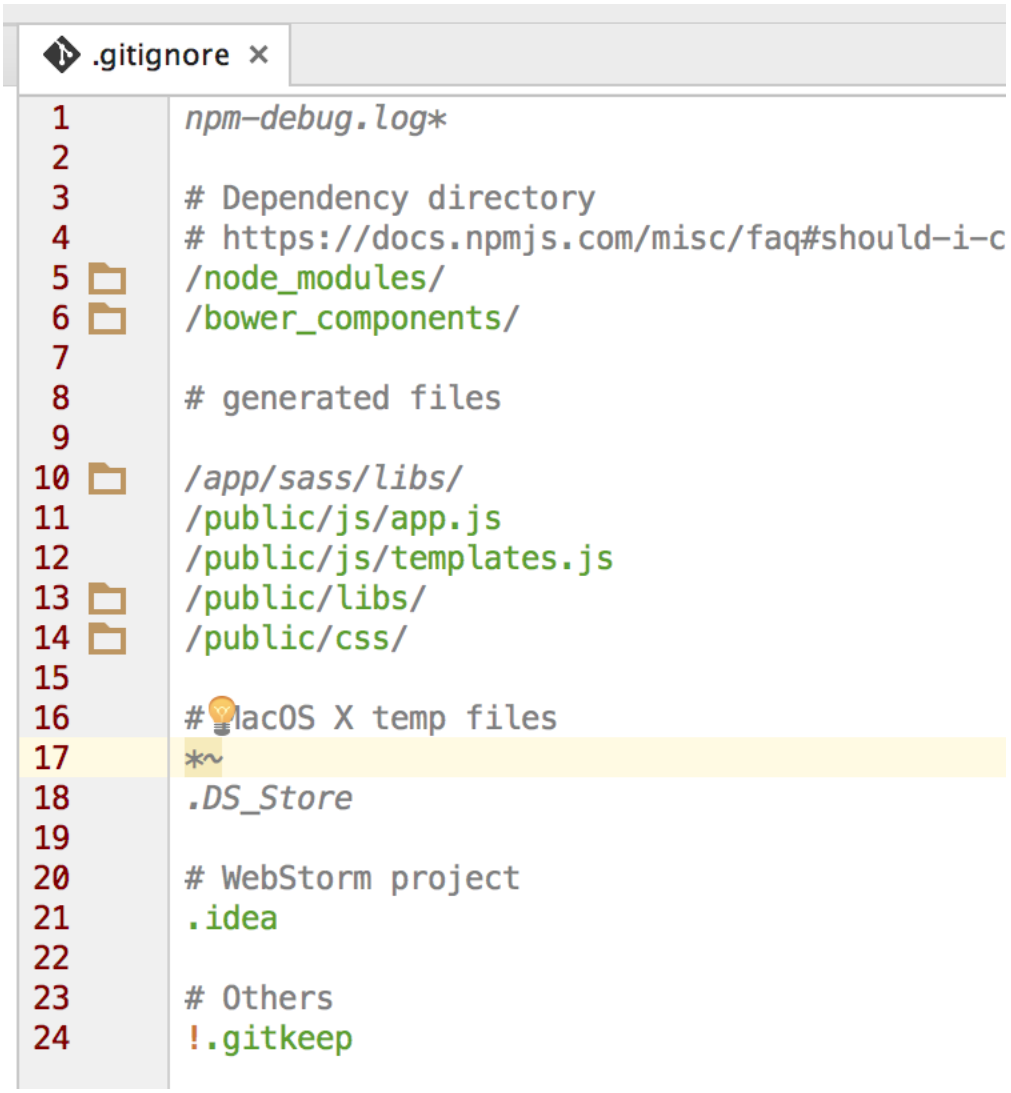
git is distributed, not centralized



Version control

Ignoring Files

.gitignore

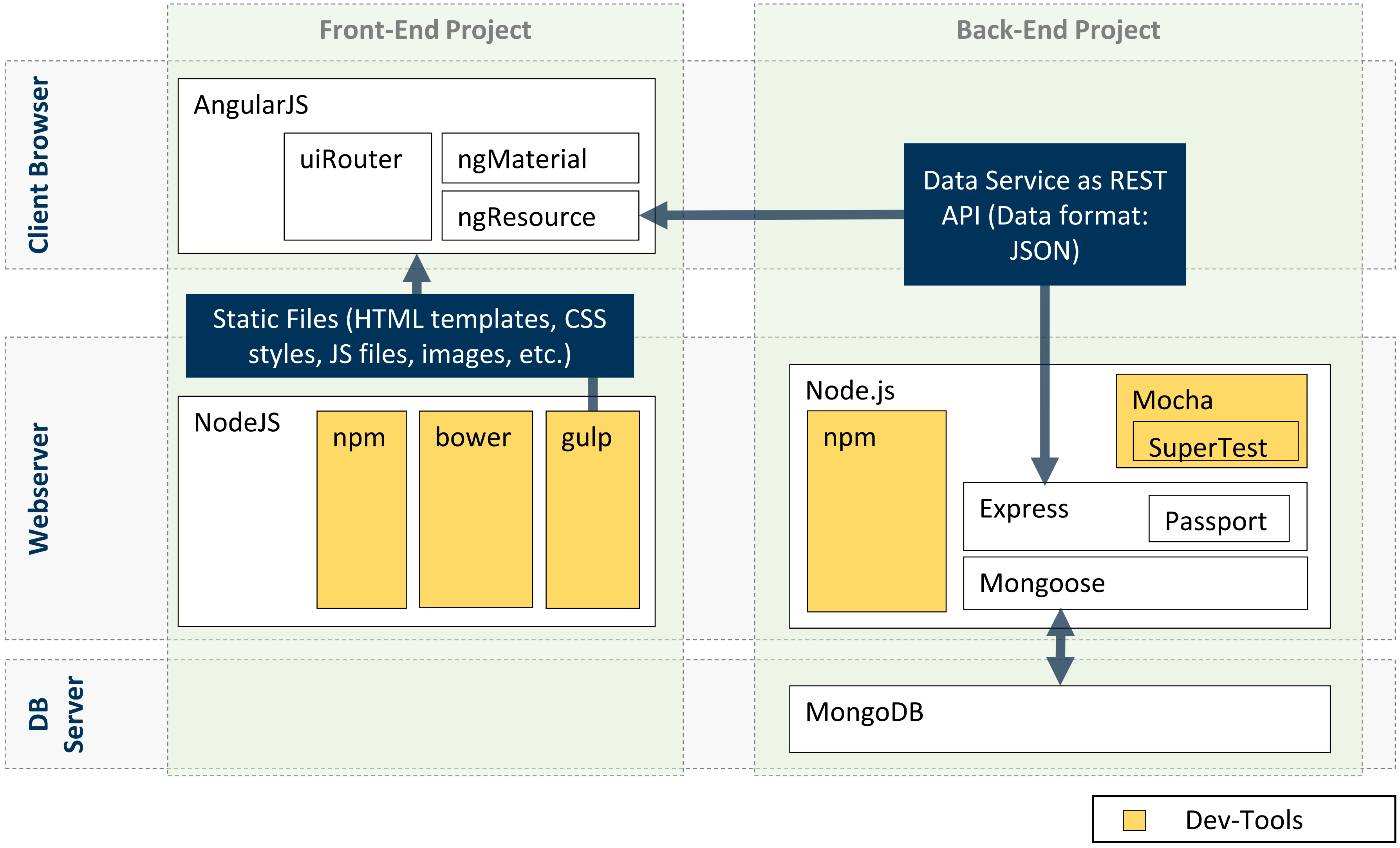


.gitignore ×

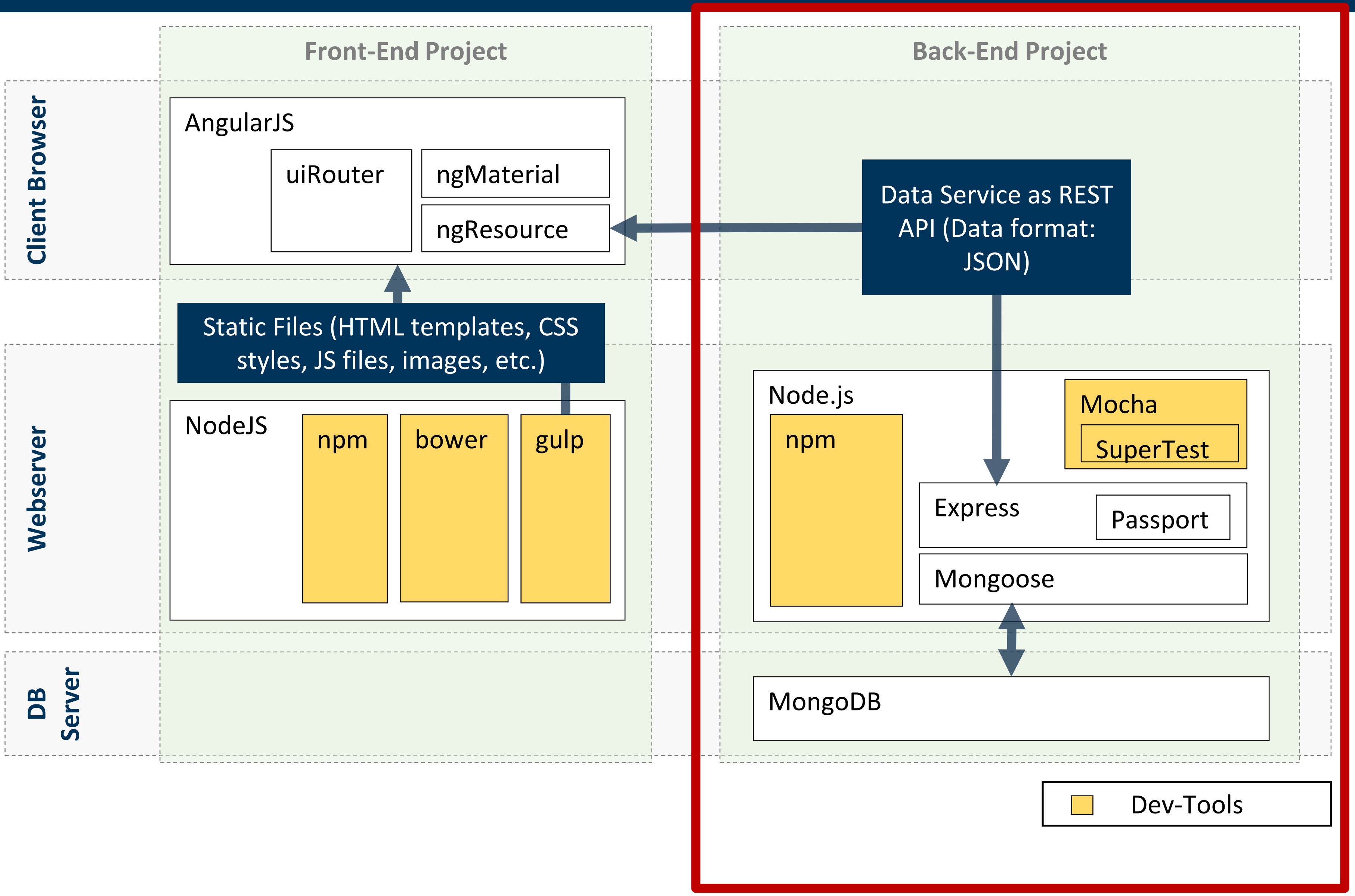
```
npm-debug.log*

# Dependency directory
# https://docs.npmjs.com/misc/faq#should-i-c
/node_modules/
/bower_components/

# generated files

/app/sass/libs/
/public/js/app.js
/public/js/templates.js
/public/libs/
/public/css/

# MacOS X temp files
*~
.DS_Store

# WebStorm project
.idea

# Others
!.gitkeep
```

Outline

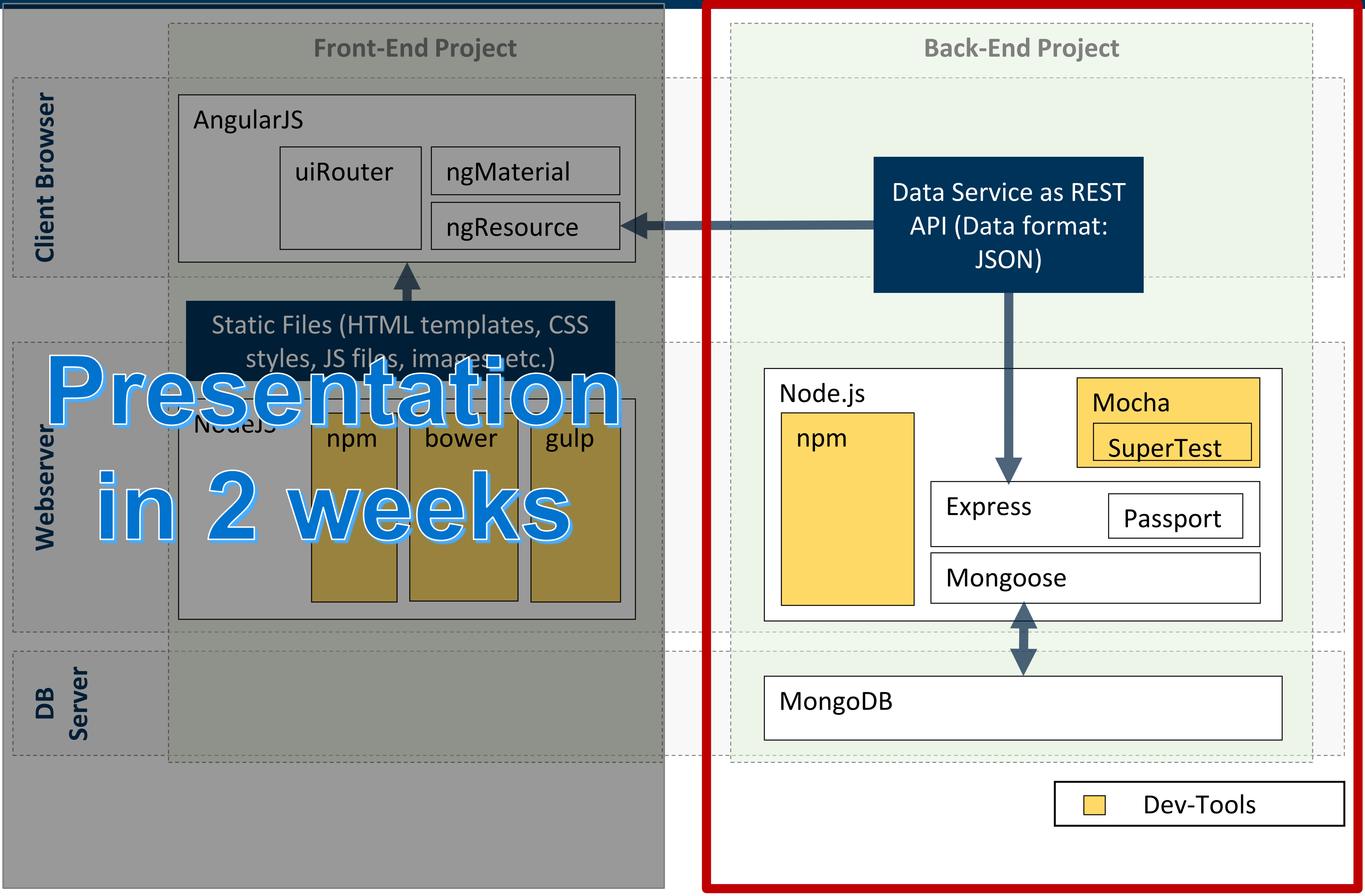
Reference architecture



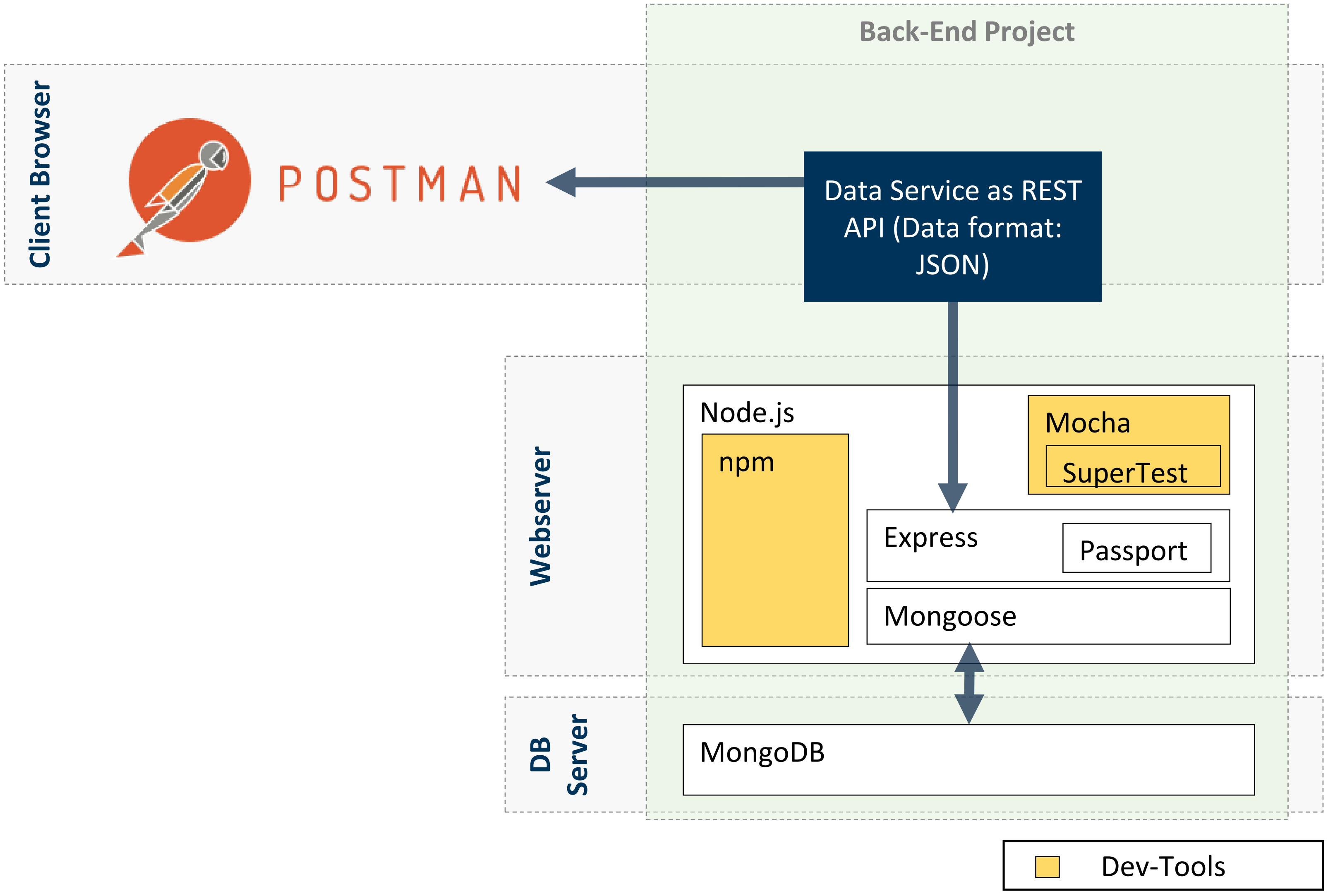
Reference architecture



Reference architecture

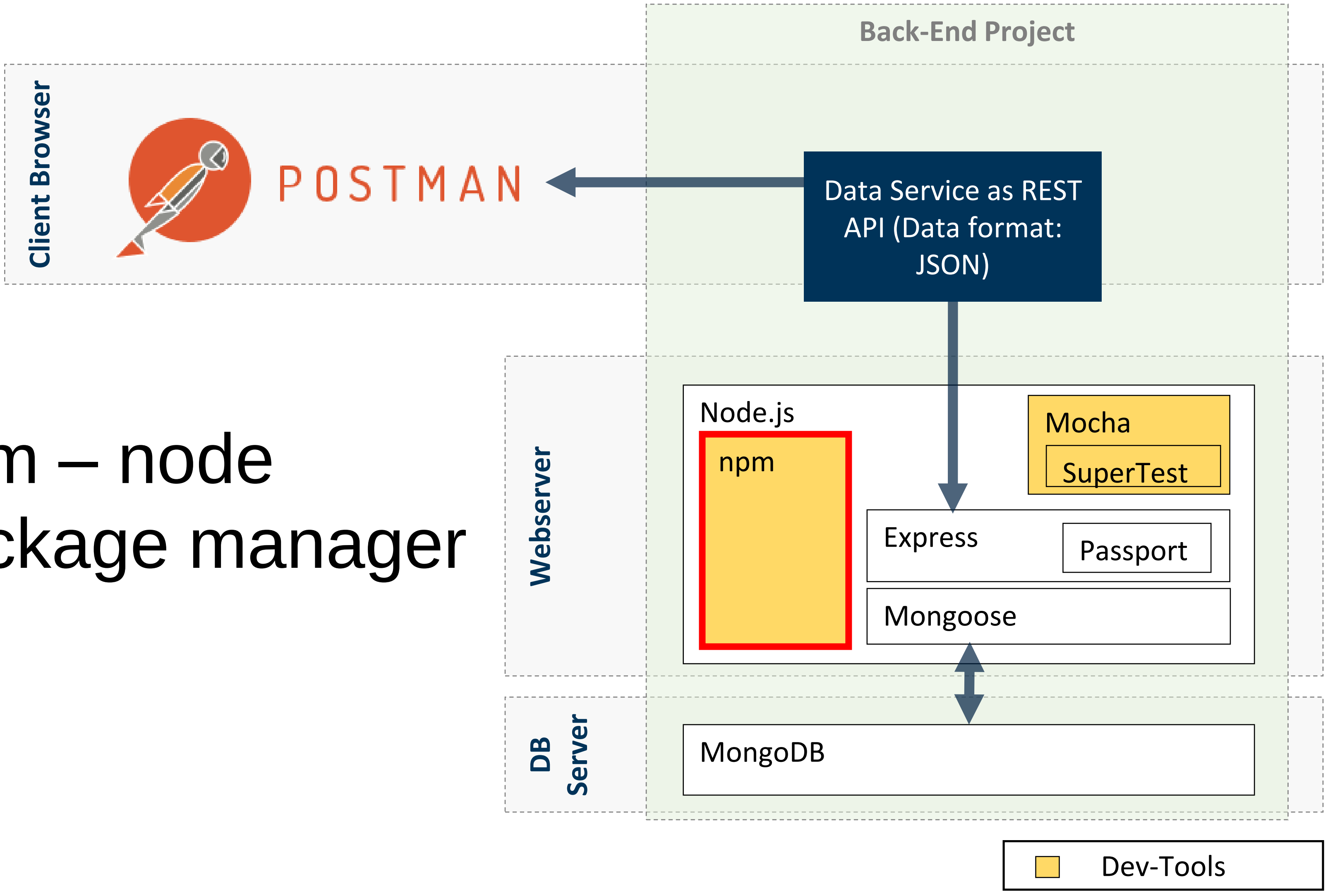


Back-end project



Node package manager

npm – node package manager



npm – node package manager

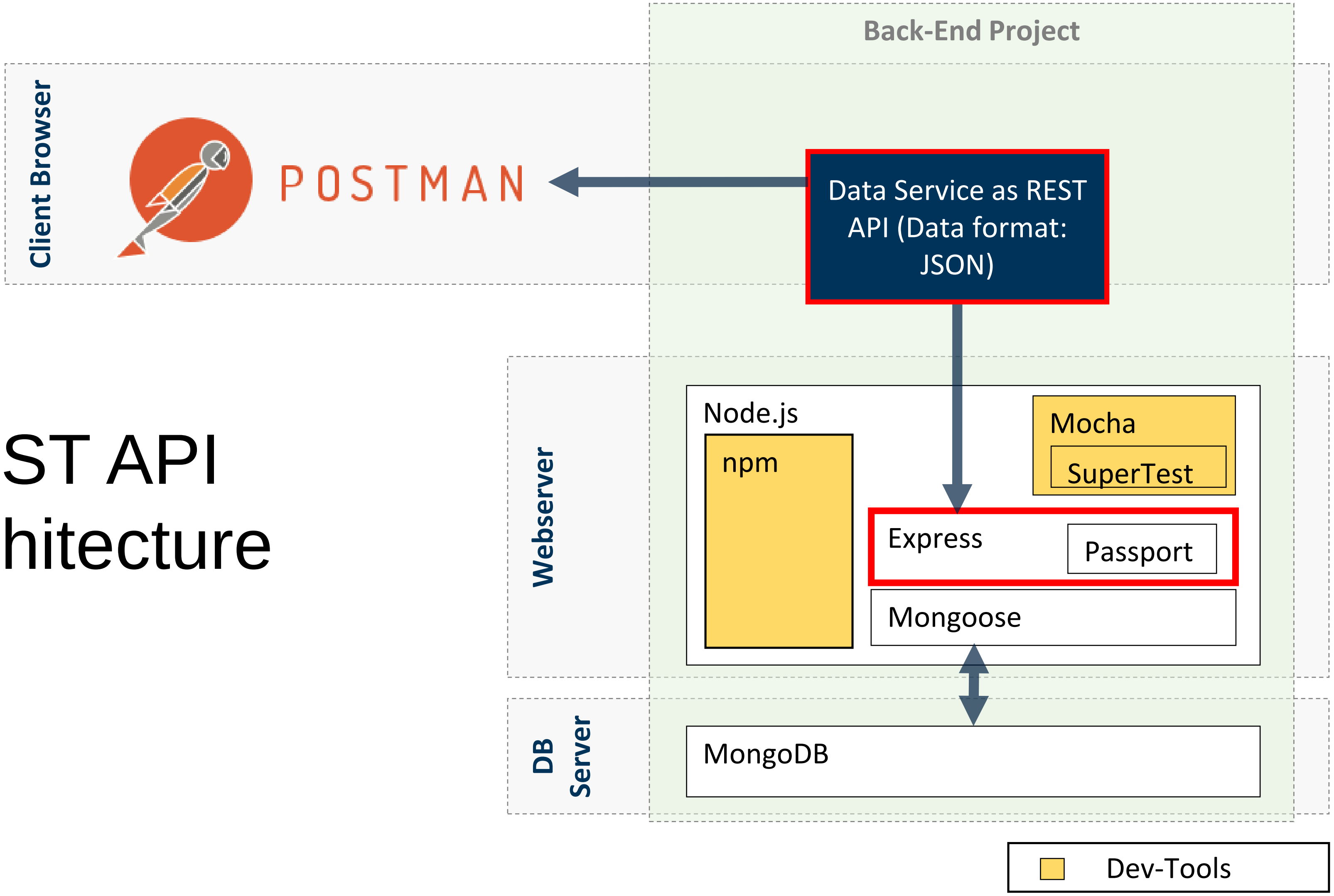
- Command line tool for handling dependencies, versions, project properties of node-js projects
- Definition of project properties and dependencies in `package.json` file
- commands are e.g.
- `npm install` – installs all dependencies from `package.json`
- `npm install <package_name> --save` – adds new npm package (e.g. **bcrypt-nodejs**, provides an API for enctypting) and saves it as dependency in `package.json`

Outline

1. Prerequisite: Version control
2. Movie app - example application
   a. Reference architecture
   b. npm – node package manager
   c. REST API architecture
   d. Database connection
   e. Authentication & Authorization with passport-js
   f. Testing witch mocha-js

REST API specification

REST API architecture



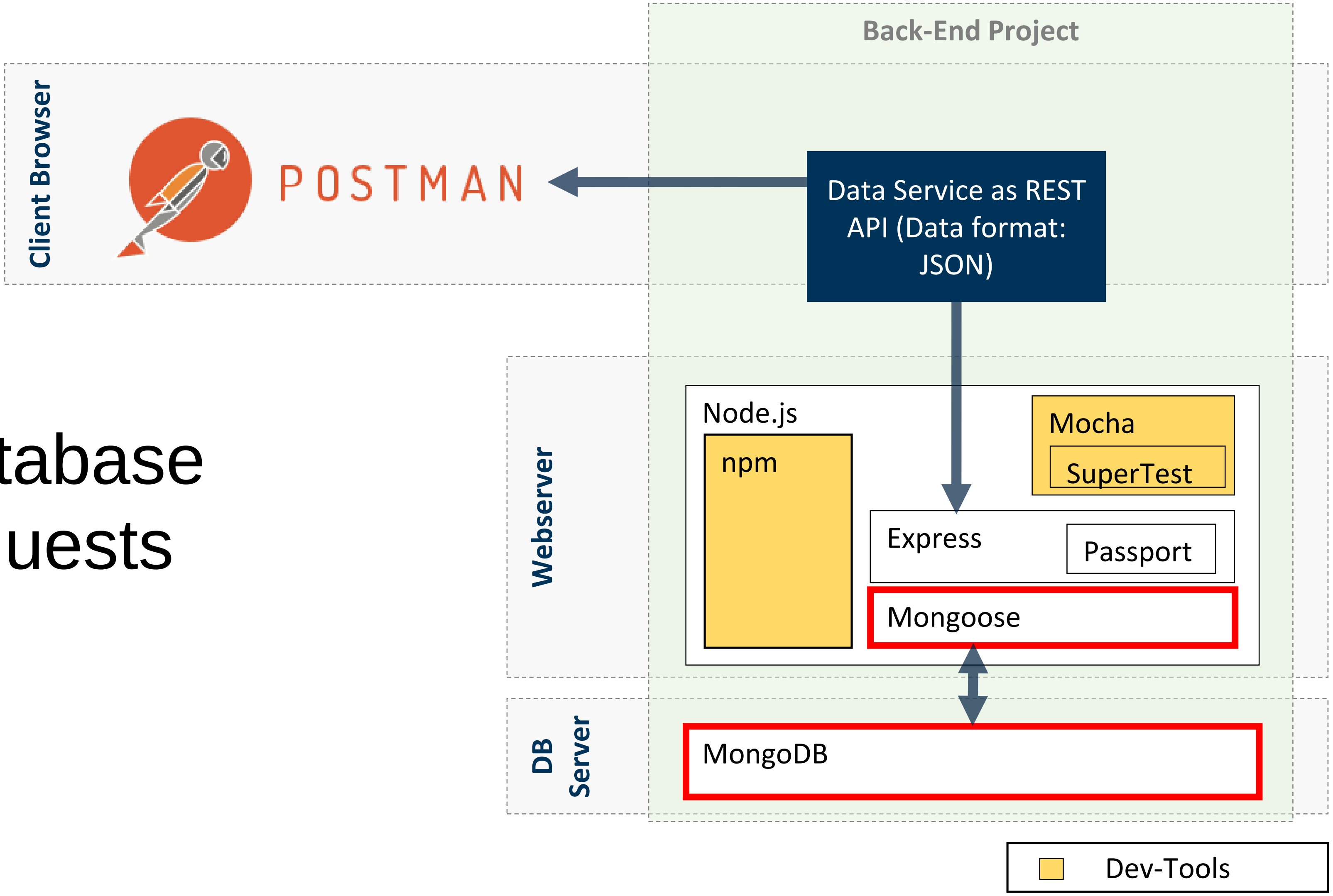
REST API specification

Http Verb (CRUD operation)	URL	Authenti cation	expected request body	expected response
GET (READ)	/movies	No	[empty]	serialized movie object
GET (READ)	/movies/{movieId}	No	[empty]	array of serialized movie objects
POST (CREATE)	/movies	Yes	serialized movie object	serialized movie object
PUT (UPDATE)	/movies/{movieId}	Yes	serialized movie object	serialized movie object
DELETE (DELETE)	/movies/{movieId}	Yes*	[empty]	[empty]

*with authorization

MongoDB

Database requests



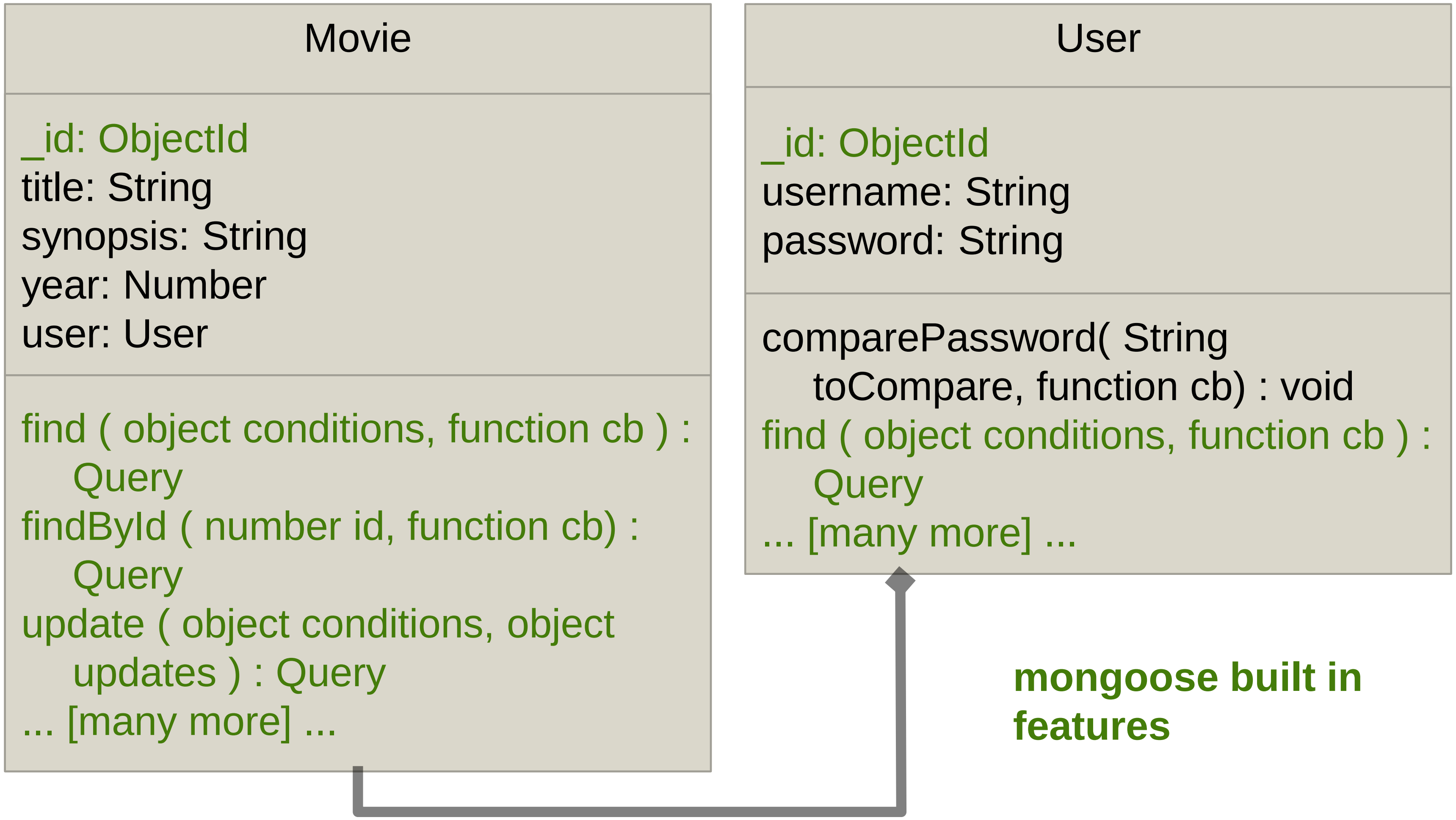
MongoDB

mongoDB® key conventions

- **Collections** (tables) do not have a fixed scheme → Scheme definitions only in your code and independent from database
- **Documents** (rows) have some built-in properties as _id (unique identifier) and _v (version number)

Models wirh Mongoose

Model definitions in mongoose



Movie
_id: ObjectId
title: String
synopsis: String
year: Number
user: User
find (object conditions, function cb) : Query
findById (number id, function cb) : Query
update (object conditions, object updates) : Query
... [many more] ...
User
_id: ObjectId
username: String
password: String
comparePassword(String toCompare, function cb) : void
find (object conditions, function cb) : Query
... [many more] ...

mongoose built in features

REST API specification

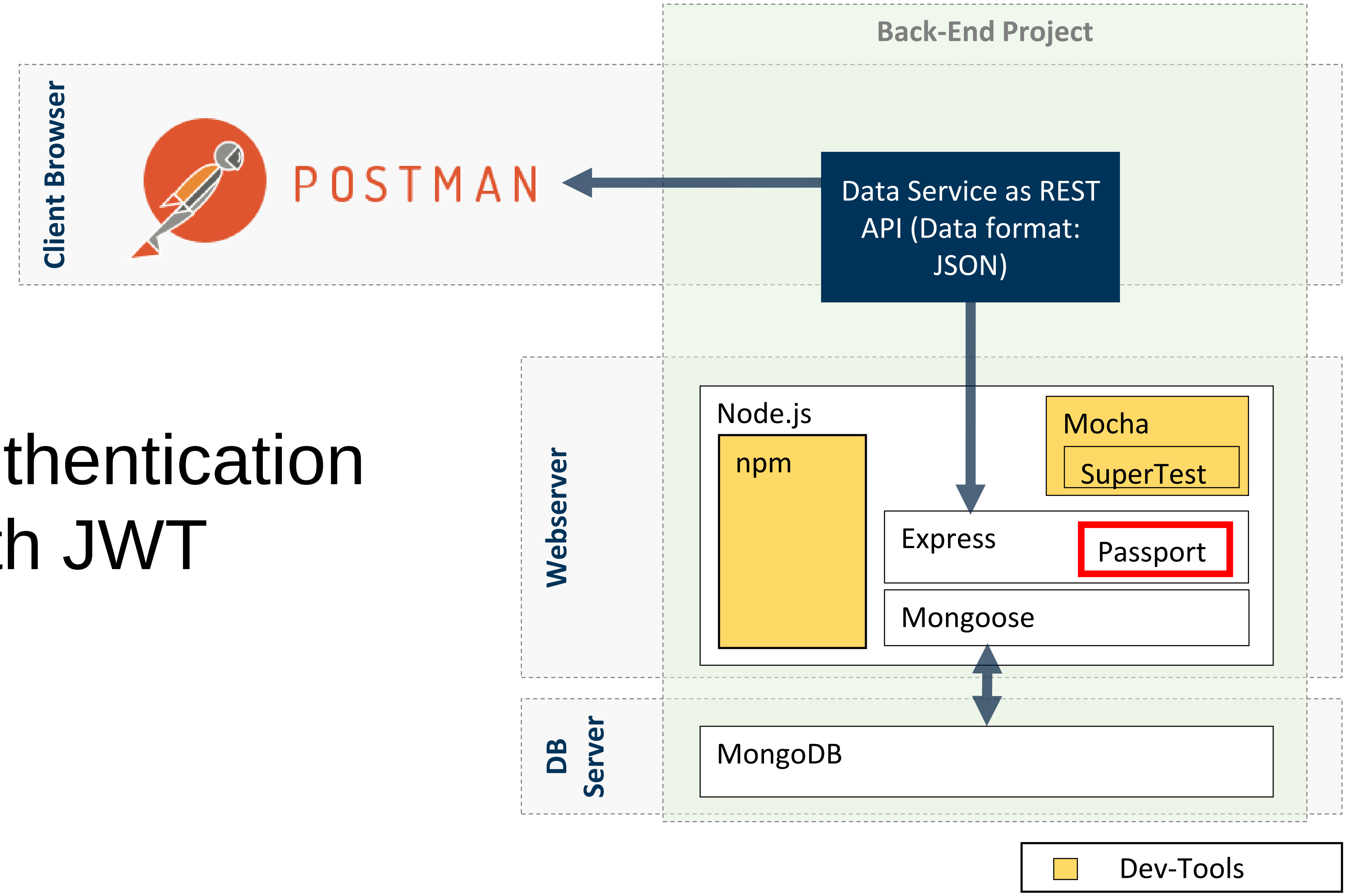
Http Verb (CRUD operation)	URL	Authentication	expected request body	expected response
GET (READ)	/movies	No	[empty]	serialized movie object
GET (READ)	/movies/{movieId}	No	[empty]	array of serialized movie objects
POST (CREATE)	/movies	Yes	serialized movie object	serialized movie object
PUT (UPDATE)	/movies/{movieId}	Yes	serialized movie object	serialized movie object
DELETE (DELETE)	/movies/{movieId}	Yes*	[empty]	[empty]

*with authorization

Outline

1. Prerequisite: Version control
2. Movie app - example application
   a. Reference architecture
   b. npm – node package manager
   c. REST API specification
   d. Database connection
   e. Authentication & Authorization with passport-js
   f. Testing witch mocha-js

Authentication with JWT

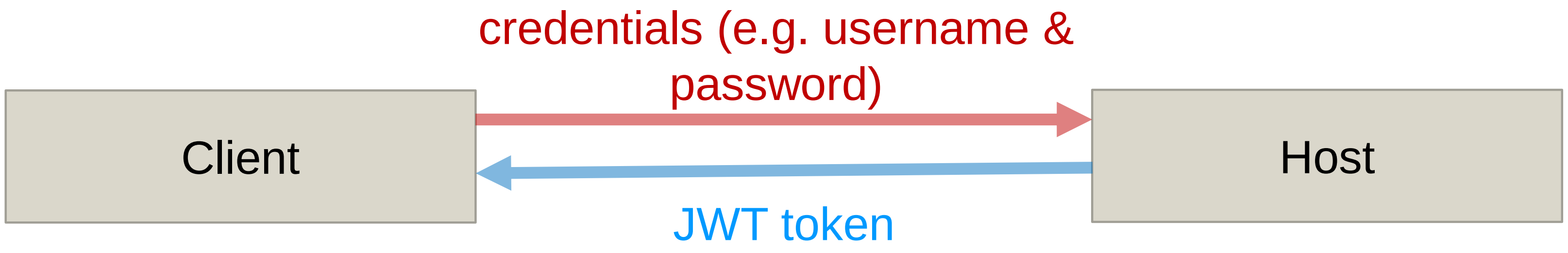


Authentication with JWT

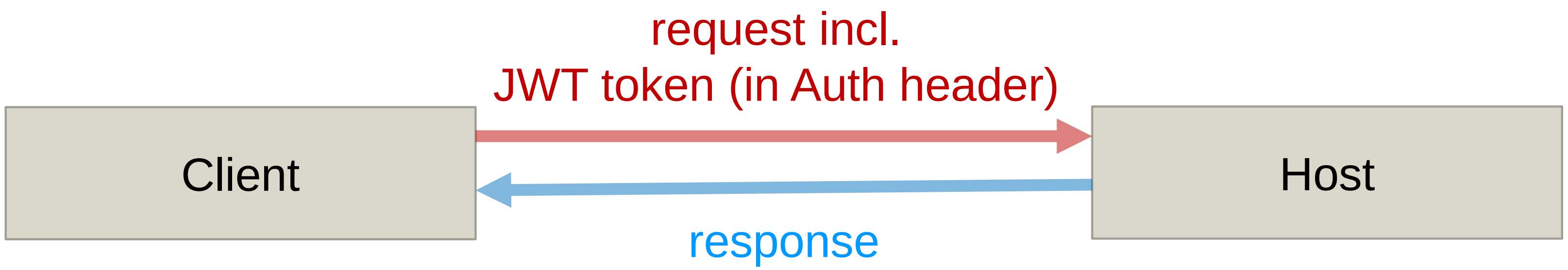
Authentication with JWT

JWT – JavaScript web tokens

Login – client sends credentials and receives JWT



Subsequent requests – client sends JWT in Authorization header



Authentication with JWT

JWT – JavaScript web tokens

Sample token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

header.payload.signature

the payload can only be **encrypted** by the host, but decrypted by everyone → It’s save to rely on payload information e.g. the user’s id

header contains meta information such as the algorithm used for encryption

Authentication with JWT

Passport

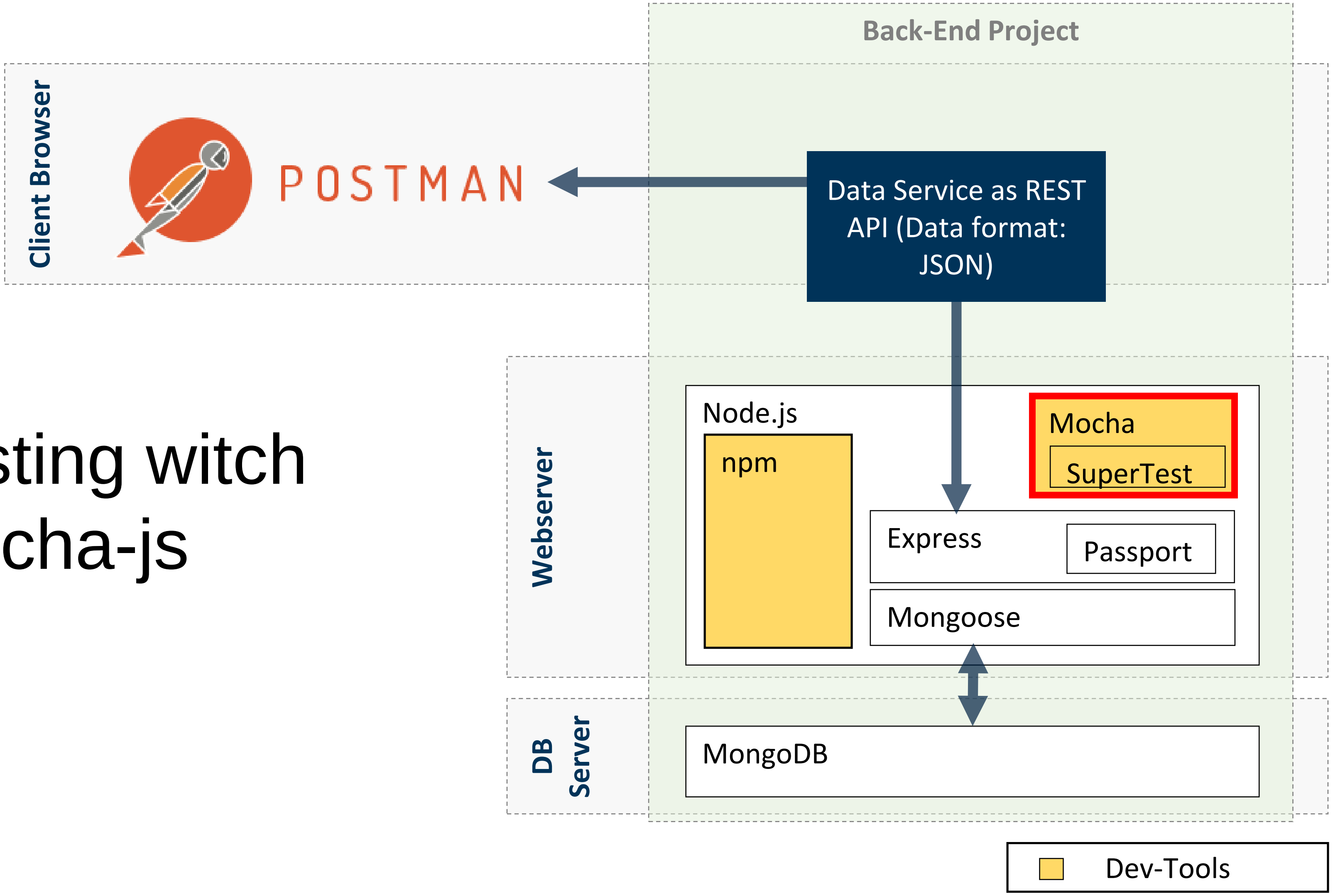
Implementation via Passport-js as **middleware**

→ middleware can be added to certain routes and executes independently before calling actual route processing function.
Examples other than passport-js:

- json **body-parser**: parses JSON-formated request
- **cors**: Modifies headers to accept cross-server requests

passport-js parses the JWT token from the auth header. If JWT is valid → injects user to the request payload → ready for e.g. further authorization

Testing witch mocha-js



Testing witch mocha-js

Testing with mocha-js

sebis

Implementations of unit tests with mocha.js

keywords:

- `describe` - describes a new collection of tests (e.g. movie lifecycle)
- `it` - defines one unit test (e.g. update movie)

hooks:

- `before` – things to do before one test collection starts (e.g. create a test user profile)
- `after` – things to do after test collection is done (e.g. clean up, delete test user record)
- `beforeEach`
- `afterEach`
- …

Web Links (1)

Web Storm IDE

https://www.jetbrains.com/student/

Create a Student Account, Download Web Storm, activate the License with your Account data

Git Version Control

http://git-scm.com/

Already included in WebSotrm, but you might need it for git tools or command line usage (e.g., SourceTree)

Bitbucket hosting

https://bitbucket.org/

Free (non-public) online hosting of your git projects

Web Links (2)

Used web technologies for development

- mongoDB https://www.mongodb.org/
- Robomongo (mongoDB GUI) https://robomongo.org/
- Postman (dev-testing your API) https://www.getpostman.com/
- node-js (run JavaScript as a program) https://nodejs.org/
- npm (node package manager) https://www.npmjs.com
- mocha (automatic testing) https://mochajs.org/

Key node modules (as in package.json)

- express http://expressjs.com/
- mongoose http://mongoosejs.com/
- passport http://passportjs.org/
- supertest https://github.com/visionmedia/supertest

Thank you for your attention! Questions?

Technische Universität München
Department of Informatics
Chair of Software Engineering for
Business Information Systems

Boltzmannstraße 3
85748 Garching bei München

Tel +49.89.289.
Fax +49.89.289.17136

wwwmatthes.in.tum.de

In 2 weeks

ANGULARJS
by Google

In 2 weeks, same place, same time

How to build your front-end application with Angular JS (and many more)