

ARCHITECTURE N-TIER NODE/QT

INTRODUCTION



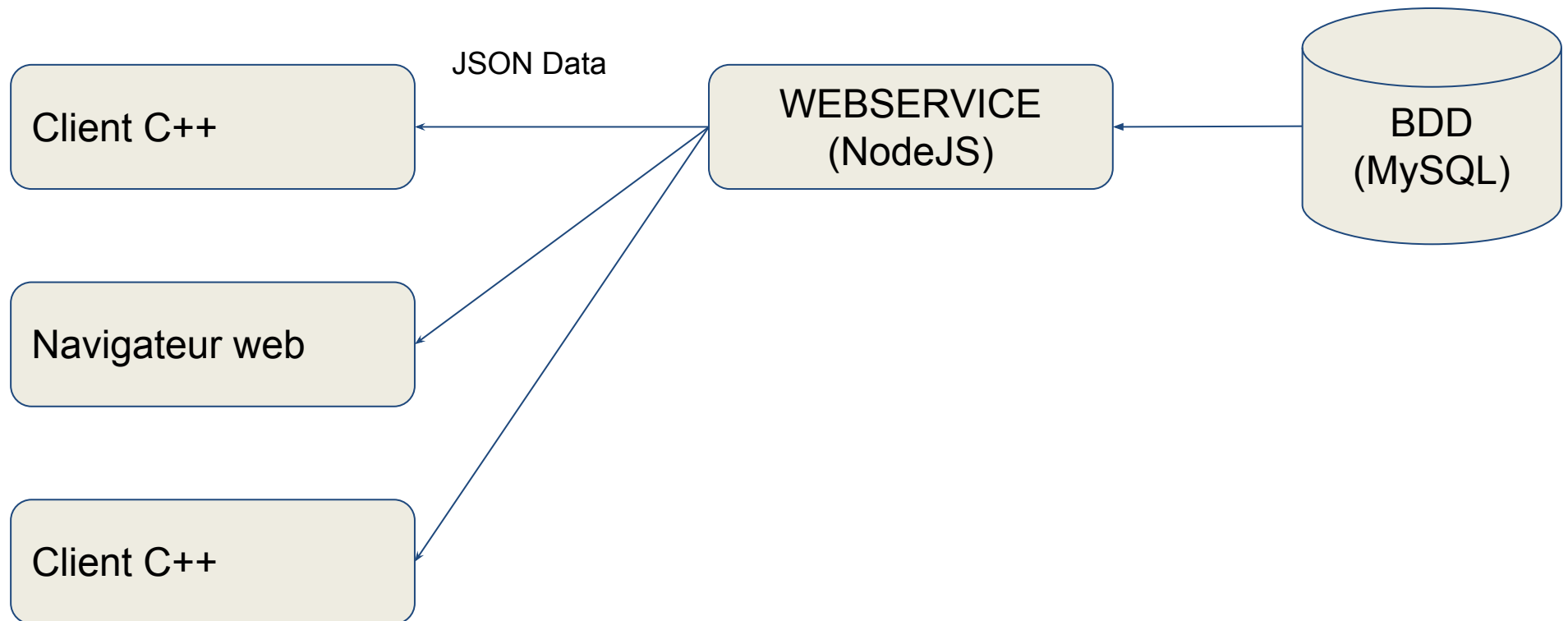
ARCHITECTURE N-TIER NODE/QT

SOMMAIRE

0. L'architecture trois-tiers	1
1. L'architecture trois-tiers par l'exemple	3
0. Présentation d'une API REST	8
2. Récupération de l'architecture	9
3. Couche 1 : BDD (MySQL)	12
4. Couche 2 : WEBSERVICE (NodeJS)	14
4. Couche 3 : CLIENT (Qt C++)	14

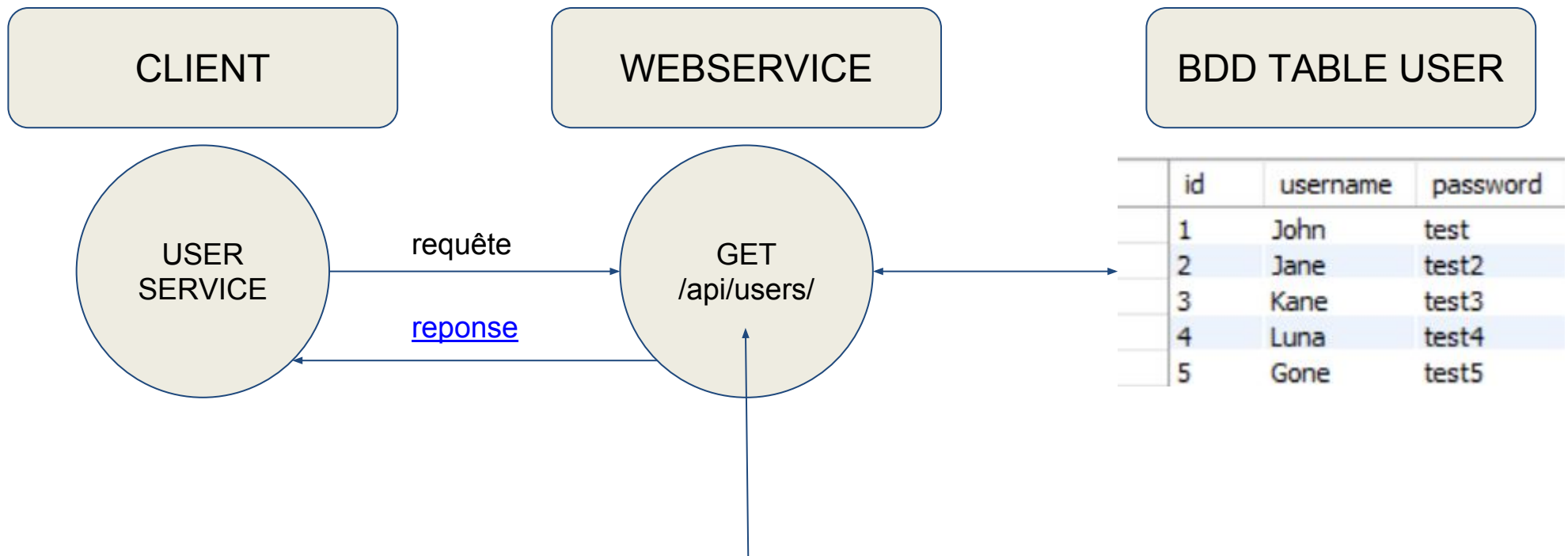
ARCHITECTURE N-TIER NODE/QT

L'ARCHITECTURE TROIS-TIERS



ARCHITECTURE N-TIER NODE/QT

L'ARCHITECTURE TROIS-TIERS PAR L'EXEMPLE



Note : Le format des url du webservice est conforme à la norme REST

ARCHITECTURE N-TIER NODE/QT

PRESENTATION DE REST

REST est simplement le nom d'une bonne pratique de notation de vos urls, celle-ci tient en 2 règles principales.

Règle 1 : L'URI comme identifiant des ressources

Liste les utilisateurs	/api/users/
Filtre et tri les utilisateurs	/api/users?filter=staff
Affiche un utilisateur	/api/users/19/
Tous les commentaires d'un utilisateur	/api/users/19/comments/




Règle 2 : Les verbes HTTP comme identifiant des opérations

Créer un utilisateur	POST /api/users/
Affiche un utilisateur	GET /api/users/19/
Supprime un utilisateur	DELETE /api/users/19/

ARCHITECTURE N-TIER NODE/QT

RECUPERATION DE L'ARCHITECTURE

git clone <https://github.com/AntiLoxy/Stargate.git>

 client	03/11/2017 16:09	Dossier de fichiers
 ressources	03/11/2017 16:04	Dossier de fichiers
 server	03/11/2017 16:12	Dossier de fichiers

ARCHITECTURE N-TIER NODE/QT

COUCHE 1 : BDD (MySQL)

Installation

Nous allons créer une base qui servira d'exemple tout au long de ce slide.

- Lancer MySQL Workbench
- Créer une base nommée dev_db
- Exécuter les requêtes du fichier `ressources/dev_db_users.sql`

Voilà, vous avez une base `dev_db` qui contient une table `users` dans laquelle se trouve plusieurs utilisateurs.

ARCHITECTURE N-TIER NODE/QT

COUCHE 2 : WEBSERVICE (NodeJS)

Installation

Nous allons procéder à l'installation ainsi qu'au lancement du webservice.

```
# cd server
```

```
# npm install \(plus d'informations sur npm\)
```

```
# npm start (le service se lance sur le port 3000)
```

<http://localhost:3000>


ARCHITECTURE N-TIER NODE/QT

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - les modules

Le webservice a sa propre architecture, ainsi chaque entité de votre application sera un module du service web.

Par exemple, une application qui doit gérer des utilisateurs ainsi que des véhicules stockés dans des parkings.

 cars	03/11/2017 16:51	Dossier de fichiers
 core	03/11/2017 16:04	Dossier de fichiers
 parkings	03/11/2017 16:51	Dossier de fichiers
 users	03/11/2017 16:04	Dossier de fichiers





Une règle est de dire que chaque table de votre base de donnée peut avoir son module.

ARCHITECTURE N-TIER NODE/QT

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - structure d'un module

Voyons de quoi se compose un module.

 business	03/11/2017 16:04	Dossier de fichiers
 controllers	03/11/2017 16:04	Dossier de fichiers
 models	03/11/2017 16:04	Dossier de fichiers
 routes	03/11/2017 16:04	Dossier de fichiers

Business	Fonctions métiers
Controllers	Fonctions appelées par les routes
Models	Définit le modèle de donnée (doit correspondre à la table associée dans la DB)
Routes	Définit les routes

WEBSERVICE PILAF.JS

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - exemple d'un module

Et maintenant son fonctionnement, prenons par exemple le module User.



WEBSERVICE PILAF.JS

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - exemple d'un module

Model

```
4  /**
5   * User Model
6   */
7  module.exports = function(sequelize, DataTypes) {
8      var User = sequelize.define('User', {
9
10         username: DataTypes.STRING,
11         password: DataTypes.STRING
12
13     });
14
15     return User;
16 };
17
```

user.model.js définit les données correspondant à la table `users` de la base de donnée.

Grâce à l'objet User nous pouvons faire toutes sortes d'opérations sur la table User de la BDD.

WEBSERVICE PILAF.JS

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - exemple d'un module

Business

```
11 exports.create = function (username, password) {
12     db.User.create({
13         username: username,
14         password: password
15     });
16 };
17
18
19 /**
20  * List of Users
21  */
22 exports.list = function () {
23     return db.User.findAll();
24 };
25
```

user.business.js contient les fonctions métiers du module users.

WEBSERVICE PILAF.JS

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - exemple d'un module

Controller

```
13 exports.create = function (req, res) {
14   |   users.create(req.body.username, req.body.password);
15   | };
16
17
18 /**
19  * List of Users
20  */
21 exports.list = function (req, res) {
22   |   users.list().then(users => {
23   |     |   res.json(users);
24   |     | });
25   | };
26
```

user.controller.js contient les fonctions appelées par les routes.

WEBSERVICE PILAF.JS

COUCHE 2 : WEBSERVICE (NodeJS)

Présentation - exemple d'un module

Route

```
7  module.exports = function (app) {  
8      var users = require('../controllers/users.controller');  
9  
10     // Users collection routes  
11     app.route('/api/users').get(users.list);  
12     app.route('/api/users').post(users.create);  
13  
14     // Single user collection routes  
15     app.route('/api/users/:userId').get(users.read);  
16     app.route('/api/users/:userId').delete(users.delete);  
17  
18     // Finish by binding the user middleware  
19     app.param('userId', users.userByID);  
20
```

user.route.js définit les routes et les fonctions du controller qui leurs sont associées.

WEBSERVICE PILAF.JS

COUCHE 3 : CLIENT (Qt C++)

Présentation

Le client doit maintenant pouvoir se connecter aux différents services REST.

Pour cela, il faut utiliser la classe `Network::Service`.

```
Service(QString name, QString host, QString api);
```

```
Service service("users", "http://localhost:3000", "api");
```


WEBSERVICE PILAF.JS

COUCHE 3 : CLIENT (Qt C++)

Présentation

```
Service service("users", "http://localhost:3000", "api");

// Récupère la ressource /api/users/
resUsers = service.getRessource();

// Récupère la ressource /api/users/:userId
resUser = service.getRessource(":userId");

// Appel le slot users() a chaque reponse d'une requete sur la ressource resUsers
connect(resUsers, SIGNAL(finished(Network::HTTPResponse)),
        this,      SLOT(users(Network::HTTPResponse)));

// Appel le slot user() a chaque reponse d'une requete sur la ressource resUser
connect(resUser,  SIGNAL(finished(Network::HTTPResponse)),
        this,      SLOT(user (Network::HTTPResponse)));

// Envoi une requête GET sur la ressource /api/users/:userId
resUser->get({{"userId", "10"}});
```

WEBSERVICE PILAF.JS

QUESTIONS ?



ARCHITECTURE N-TIER NODE/QT

NPM : LE PACKAGE MANAGER DE NODE.JS

```
# npm install <package-name> (installation globale)
# npm install <package-name> --save (installation locale)
# npm uninstall <package-name> (suppression globale)
# npm uninstall <package-name> --save (suppression locale)

# npm install (cherche le fichier packages.json et installe)
```

▼ 0:

id: 1
username: "John"
password: "test"

▼ 1:

id: 2
username: "Jane"
password: "test2"

▼ 2:

id: 3
username: "Kane"
password: "test3"

▼ 3:

id: 4
username: "Luna"
password: "test4"

▼ 4:

id: 5
username: "Gone"
password: "test5"