Présentation Finale

Open Source Framework : Team 1

https://github.com/falkin/OSFGamificationProject

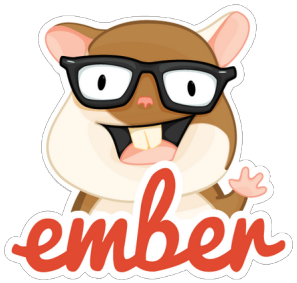
MSE | MASTER OF SCIENCE IN ENGINEERING

V. Bourqui, D. Crausaz, S. Dunand & M. Fauvet

Plan

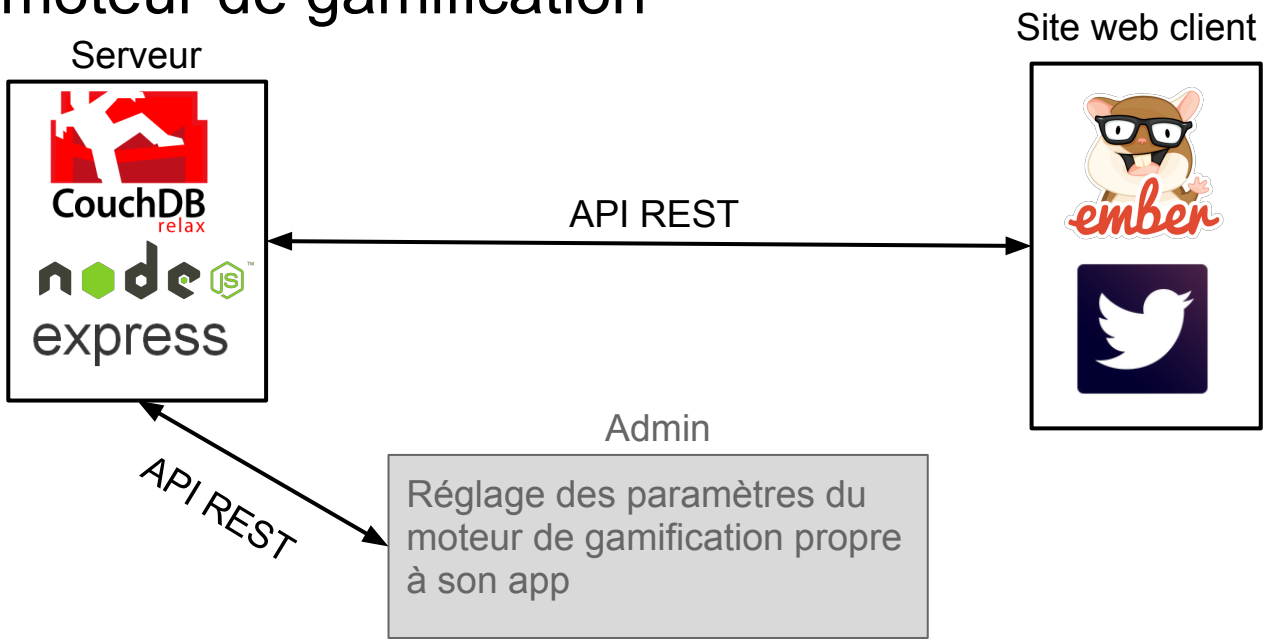
- Introduction
- Serveur
  - CouchDB
  - node.js
  - express.js
- Client
  - ember.js
  - bootstrap
- Démonstration
- Conclusion
- Questions

express



Introduction

- Moteur de gamification
  - **Serveur**: moteur de gamification accessible via une API REST
  - **Client**: site web utilisant le moteur de gamification
  - **Administrateur**: admin du site web client, et de son moteur de gamification



CouchDB

Performances

- Comparaison avec MySQL
- Un document par objet ou document par type d'objet ?

Mesures dans node.js

Modules : mysql.2.0.0
nano
async (async.series - exécution des tâches en séquence)

Serveurs CouchDB et MySQL en local

CouchDB

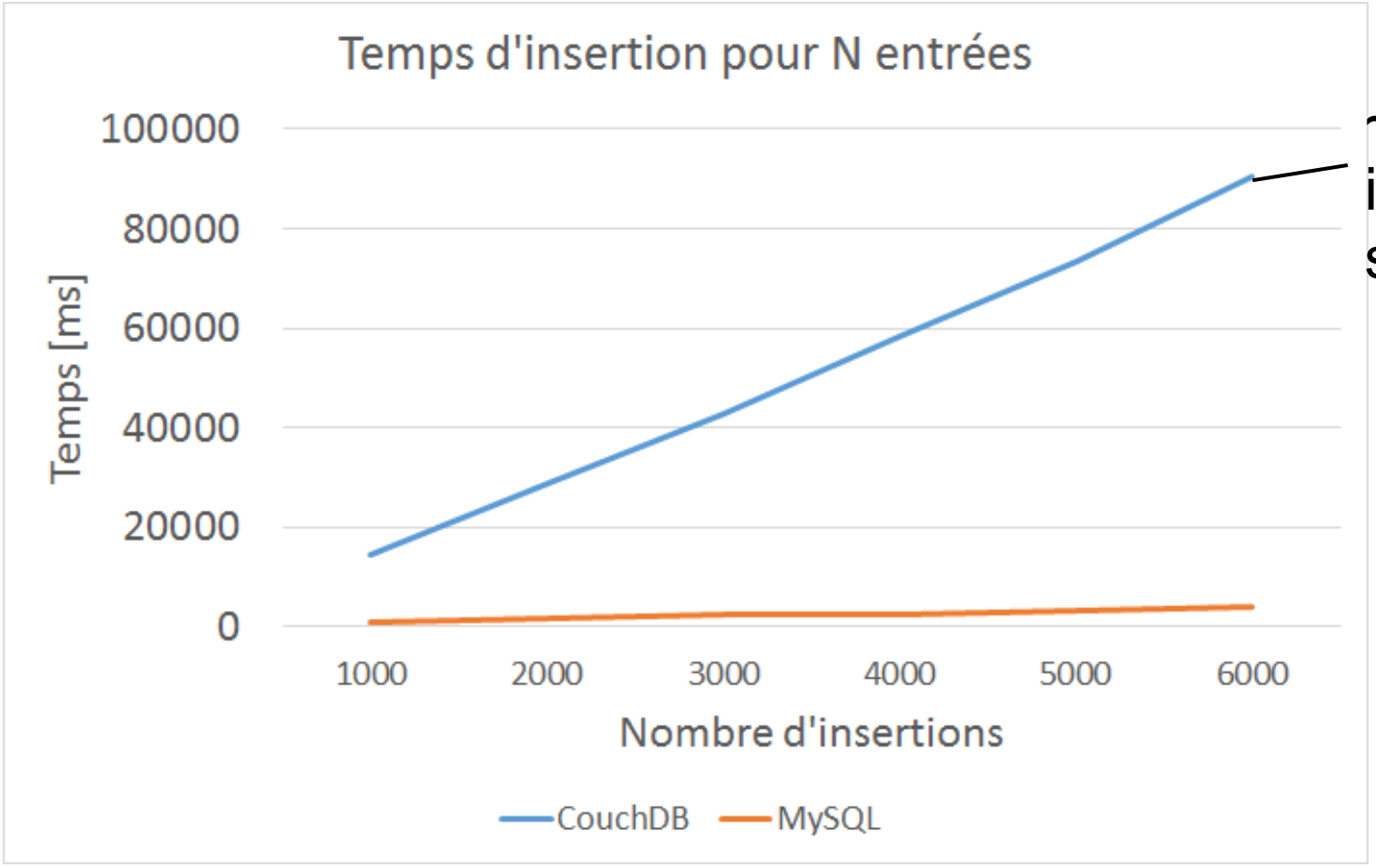
MySQL

```
INSERT INTO user (id, app_id, ...) VALUES (2, 3, ...);
```

CouchDB

```
POST   http://127.0.0.1:5984/osf_database
       {"_id": 2,
        "app_id: 3,
        "type" : "user",
        ...}
```

CouchDB



~15 insertions/ secondes

CouchDB relax

CouchDB

- Bulk (modifier/créer plusieurs documents en une seule requête)

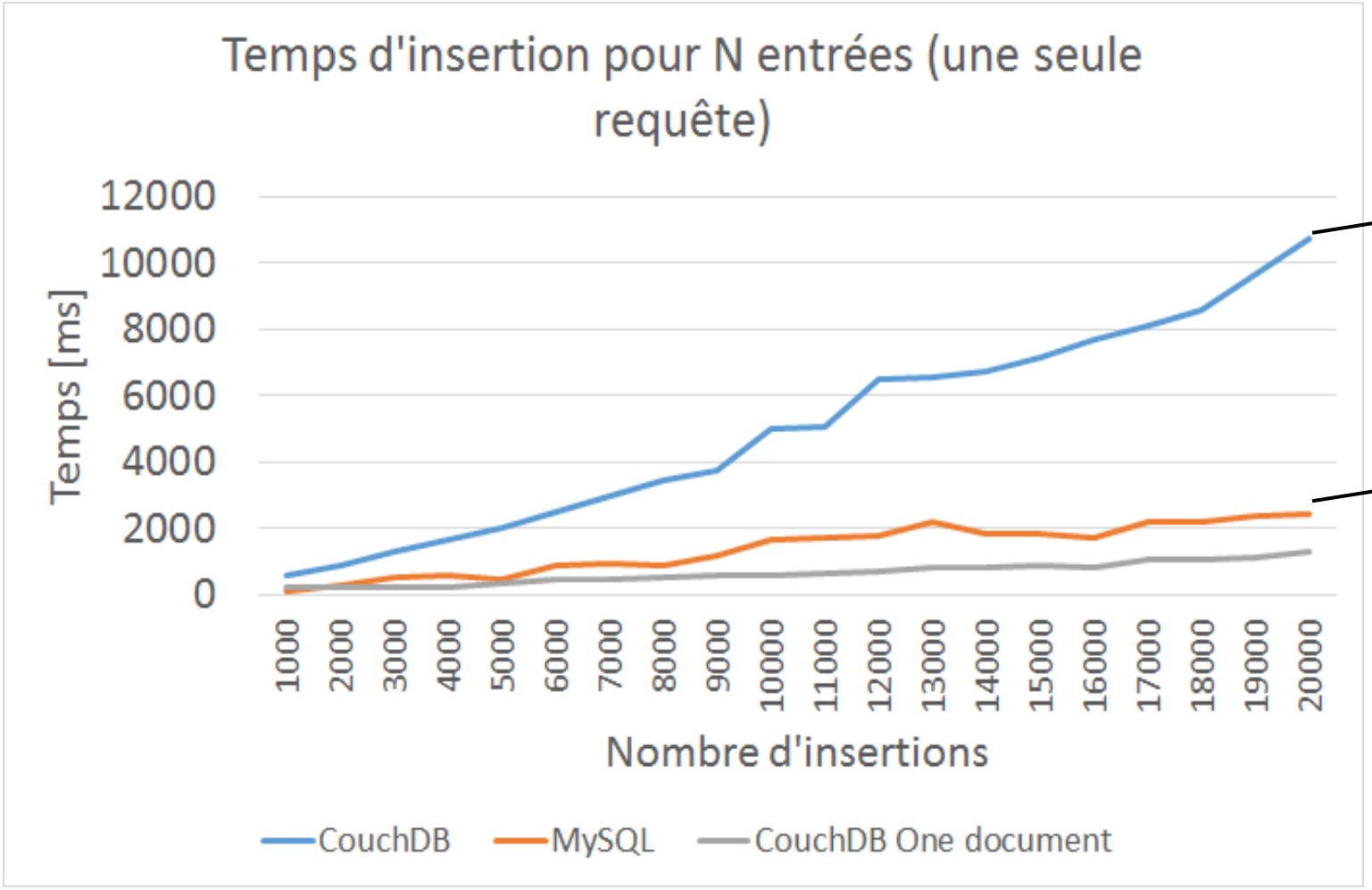
MySQL

```
INSERT INTO user (id, app_id, ...) VALUES (2, 3, ...), (3, 3, ...);
```

CouchDB

```
POST   http://127.0.0.1:5984/osf_database/_bulk_docs
        {"docs":[{"_id": 2,"app_id: 3,"type":"user",...},
                 {"_id": 3,"app_id: 3,"type":"user",...}]}
```

CouchDB



CouchDB 35x plus rapide avec bulk

MySQL 5x plus rapide

CouchDB relax

CouchDB

MySQL

SELECT * FROM user WHERE id=2 and app_id=3

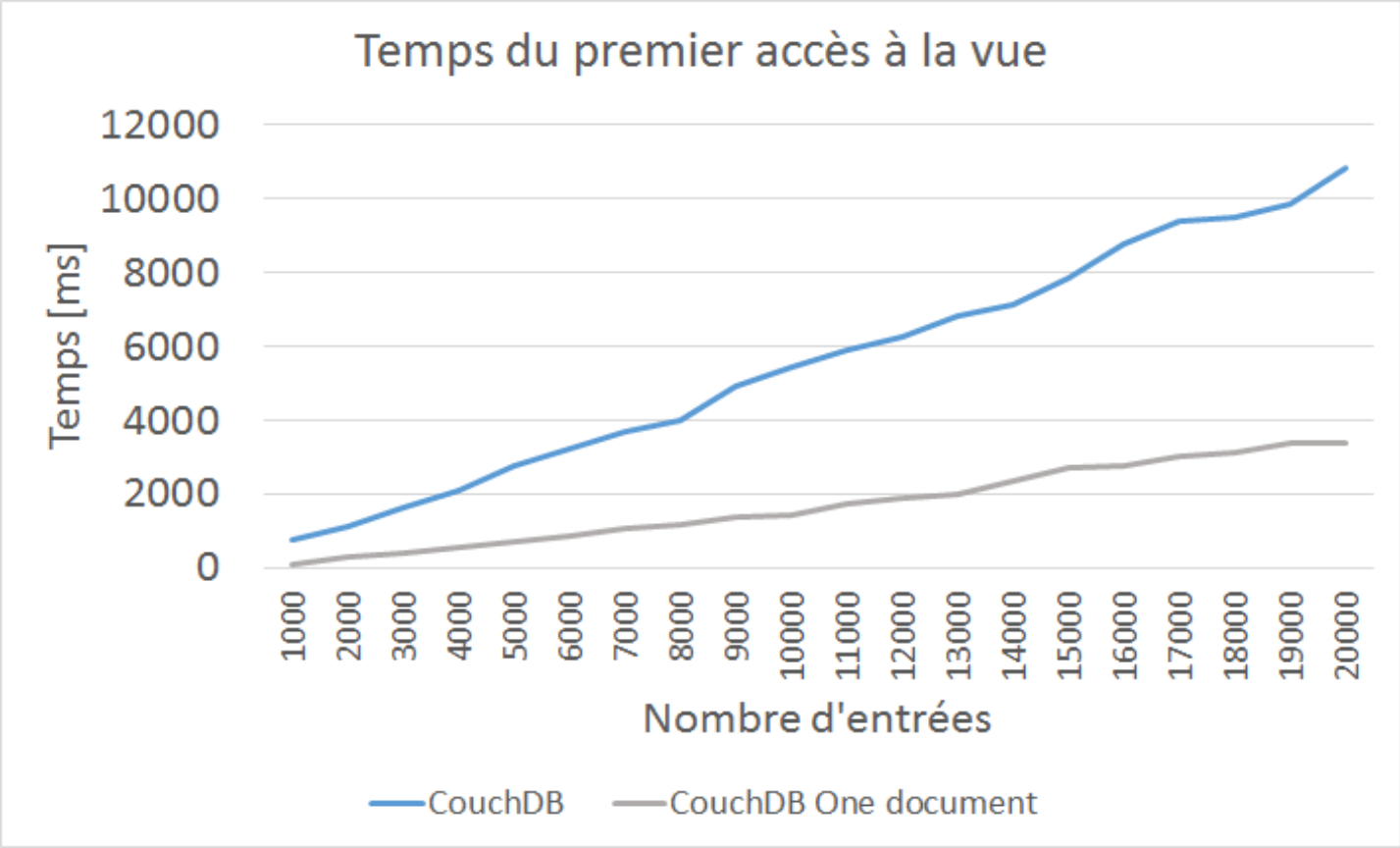
CouchDB

GET http://127.0.0.1:5984/osf_database/_design/users/_view/allUsersByID?key=2

```
//view
allUsersByID" : {
        map : function(doc) {
                if (doc.type == "user") {
                        emit(doc._id, doc);
                }
        }
}
```

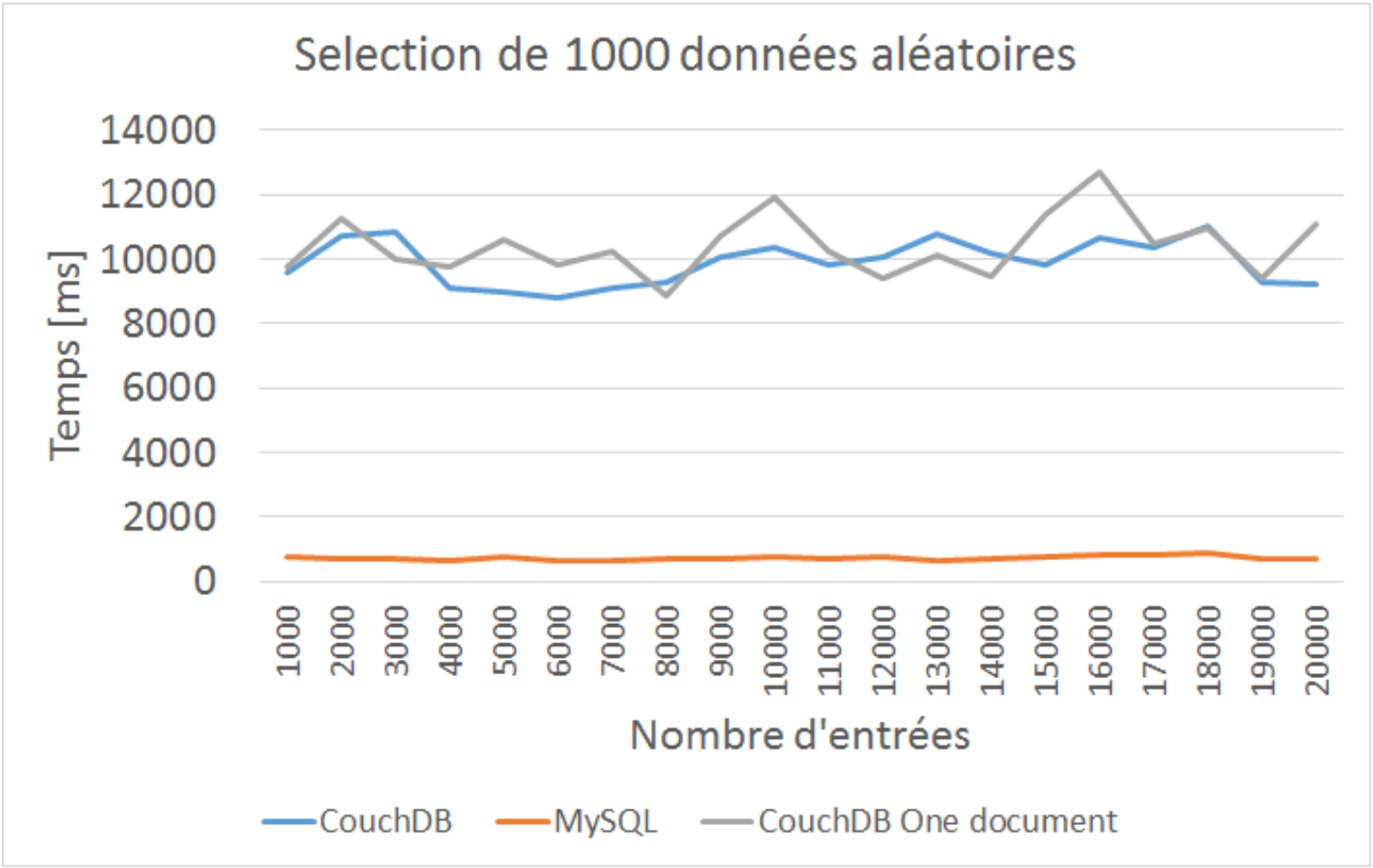
CouchDB

- Le premier accès la vue est long car il y a la génération de l'index



CouchDB
relax

CouchDB



CouchDB
relax

node.js

- Plate-forme construite sur l'exécution JavaScript pour construire facilement, des applications réseau, surtout des serveur web.
- Idéale pour les applications en temps réel.
- Intègre une librairie serveur HTTP

node.js

- lèger et simple d'utilisation
- Nombreux modules
- Bonne documentation

- Trop de modules pas assez réglémenté.
- Erreurs remontées pas toujours compréhensibles.

node.js

- **Express** : Web application framework
- **requirejs** : Structure
- **validator** : valider le contenu
- **Nano** : liaison avec couchDB

node.js - nano

Définition de la vue :

```
engine.insert(
        {  "shows":
            { "allByGameEngineID": function(doc, req)
                    {
                            return { body: JSON.stringify({ gameEngine : doc }) };
                    }
             }
            , "updates":
                        }
      }, '_design/gameEngines'
```

node.js - nano

Sélectionner des informations dans couchDB:

```
// Select a game engine
selectGameEngine = function(req, res) {
    engine.show('gameEngines', 'allByGameEngineID', req.params.appid, function(err, doc) {
        if (err) {
            sendResponse.sendErrorsDBError(res, err);
        } else {
            if (doc.gameEngine == null) {
                sendResponse.sendErrorsNotFound(res, "GameEngine not found");
                return;
            }
            sendResponse.sendObject(res, gameEngineResponse(doc.gameEngine));
        }
    });
```

node.js - nano

Autre vue :

```
{ "views":
        { "allObjcetsByAppID":
            {
                map: function (doc) {
                    emit([doc.appID], doc);
                }
            }
        }
```

express.js

Création de notre serveur et configuration.

```
 var app = express();
app.configure(function() {
        app.set('port', process.env.PORT || expressVariables.PORT);
        app.set('ip', expressVariables.IP);
        app.use(express.bodyParser());
        app.use(expressVariables.allowCrossDomain);
        app.use(app.router);
        app.use(express.static(expressVariables.__dirname + '/public'));
        app.use(logErrors);
        app.use(clientErrorHandler);
        app.use(errorHandler);
    });
http.createServer(app).listen(app.get('port'),app.get('ip')...
```

express

express.js

Définition des URLs d'écoute.

```
// game engine call and manage
app.post("/admin/game_engine",adminGameEngine.createGameEngine);
app.get('/admin/game_engine/:appid',adminGameEngine.selectGameEngine);
app.put('/admin/game_engine/:appid',adminGameEngine.updateGameEngine);
app.delete('/admin/game_engine/:appid',adminGameEngine.deleteGameEngine);
```

Tests automatiques

- **Travis CI**: service d'intégration continue connecté à Github
- **vows**: BDD pour Node
- **api-easy**: syntaxe permettant de générer facilement des tests vows pour une API REST

vows language for a POST

```
var request = require('request'),
    vows = require('vows'),
    assert = require('assert');

vows.describe('your/awesome/api').addBatch({
  "When using your awesome api": {
    "and your awesome resource": {
      "A POST to /awesome": {
        topic: function () {
          request({
            uri: 'http://localhost:8080/awesome',
            method: 'POST',
            body: JSON.stringify({ test: 'data' }),
            headers: {
              'Content-Type': 'application/json'
            }
          }, this.callback)
        },
        "should respond with 200": function (err, res, body) {
          assert.equal(res.statusCode, 200);
        },
        "should respond with ok": function (err, res, body) {
          var result = JSON.parse(body);
          assert.equal(result.ok, true);
        },
        "should respond with x-test-header": function (err, res, body) {
          assert.include(res.headers, 'x-test-header');
        }
      }
    }
  }
}).export(module);
```

same POST with api-easy

```
var APIeasy = require('api-easy'),
    assert = require('assert');

var suite = APIeasy.describe('your/awesome/api');

suite.discuss('When using your awesome API')
     .discuss('and your awesome resource')
     .use('localhost', 8080)
     .setHeader('Content-Type', 'application/json')
     .post('/awesome', { test: 'data' })
       .expect(200, { ok: true })
       .expect('should respond with x-test-header', function (err, res, body) {
         assert.include(res.headers, 'x-test-header');
       })
     .export(module);
```

ember.js

- Framework MVC javascript basé sur Sproutcore (projet Apple)
- Utilisé pour créer des applications web de type "single page web application"
- S'appuie et intègre les librairies Handlebars et Metamorph

ember.js

Fonctionnellement riche:

- Data Binding
- Computed Properties
  - Permet de traiter une fonction comme une propriété
- Auto-Updating templates
  - Avec handlebars et metamorph

Peu de documentations

Pas mal de bugs non résolus

Problèmes de versioning

ember.js

Data Binding:

- Lie 2 propriétés de telle façon que si l'une change l'autre est mise à jour

```
App.Post = DS.Model.extend({
    title : DS.attr('string'),
    author : DS.belongsTo('App.Person'),
    intro : DS.attr('string'),
    extended : DS.attr('string'),
    publishedat : DS.attr('date')
});
```

```
<script type="text/x-handlebars" id="post/_edit">
        <p>Title : {{view Ember.TextField valueBinding='title'}}</p>
        <p>Intro : {{view Ember.TextArea valueBinding='intro'}}</p>
        <p>Text : {{view Ember.TextArea valueBinding='extended'}}</p>
</script>
```

ember.js

Intègre nativement handlebars:

- Un langage de templating en javascript

```
person =  [
 {
 "name": "John Cobra",
 "posts": [1]
}
 {
 "name": "Alain",
 "posts": [2]
 }
]
```

```
<ul>
{{#each person}}
    <li> Hello, {{name}} ! </li>
{{/each}}
</ul>
```

```
<ul>
    <li> Hello, John Cobra ! </li>
    <li> Hello, Alain ! </li>
</ul>
```

ember.js

WARNING: EMBER-DATA IS A WORK IN PROGRESS AND UNDER RAPID DEVELOPMENT.
USE WITH CAUTION!!!

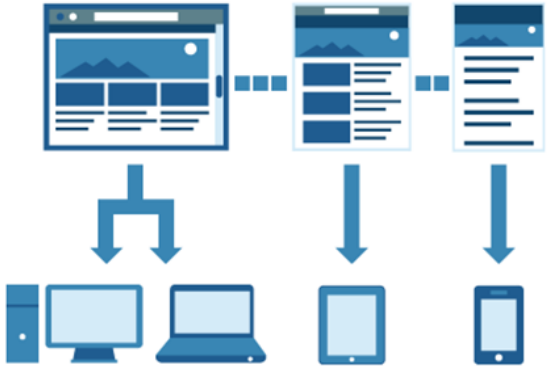
- Central data store
  - peut être configuré avec plusieurs Adapter
  - Deux adapter
    - RESTAdapter
    - ElasticSearchAdapter

```
App.Store = DS.Store.extend({
  revision: 12,
  adapter: DS.ElasticSearchAdapter.create({
        url: 'http://localhost:9200'
   })
});
```

```
App.Store.registerAdapter('App.User',
   DS.RESTAdapter.extend({
        url : "http://127.0.0.10:3000/app/{app_id}",
        mappings : {
          user : App.User
   }
}));
```

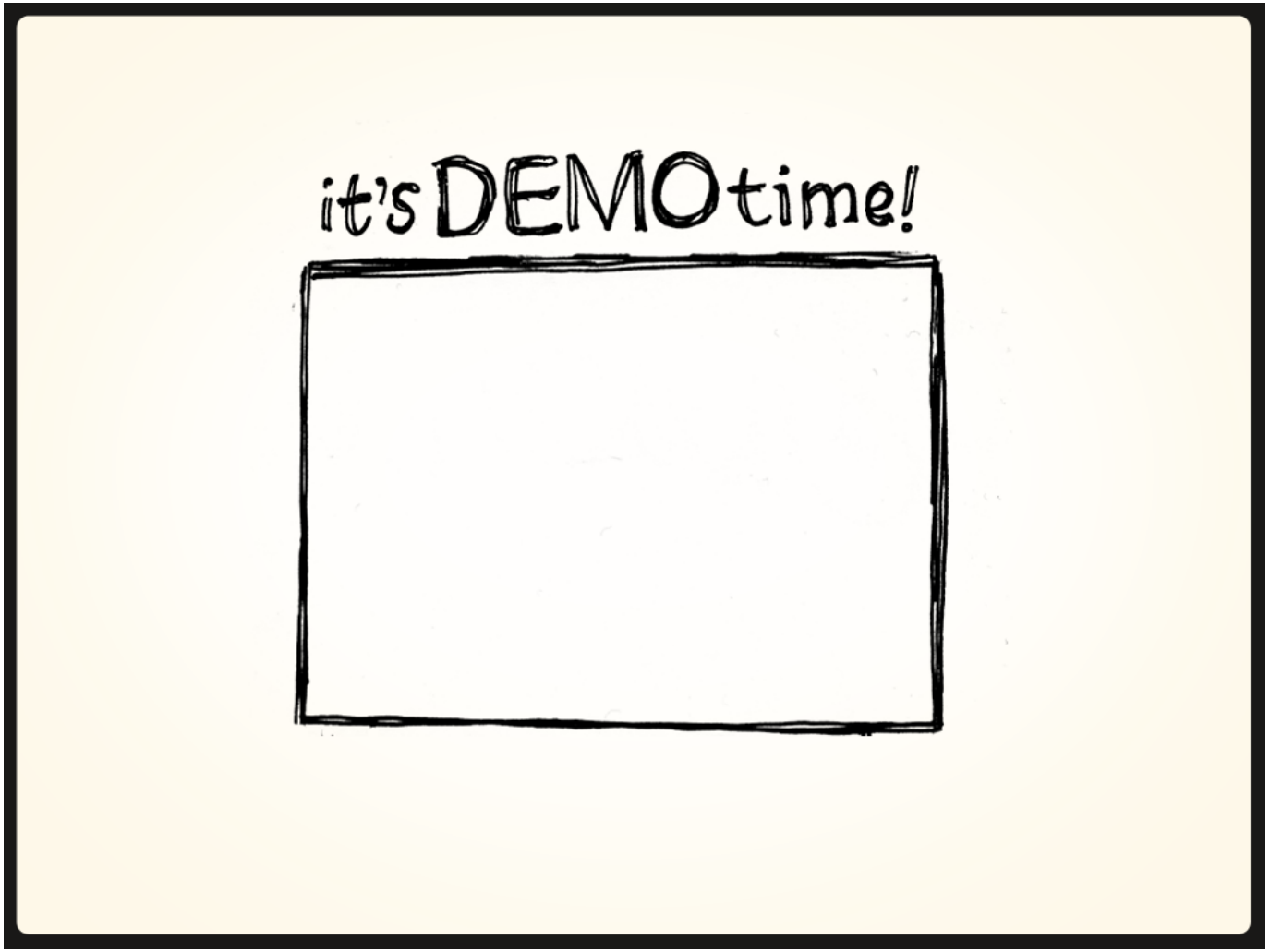
bootstrap

- Template complet
- Responsive design



- Compatible avec beaucoup de navigateur (IE7!)
- Système de grille à 12 colonnes
- Formulaires, menu dropdown, et bien plus

Démonstration



Conclusion

- Développement aisé et efficace du serveur
  - couchdb, node et express mature
- On ne peut pas en dire autant du client
  - ember pas au point, et pas assez documenté
- Travail d'équipe efficace

Références

- CouchDB http://couchdb.apache.org/
- Node.js http://nodejs.org/
- Express.js http://expressjs.com/
- Ember.js http://emberjs.com/
- Bootstrap http://twitter.github.io/bootstrap/
- Travis CI https://travis-ci.org/
- Vows http://vowsjs.org/
- API-easy https://github.com/flatiron/api-easy

Questions