GlassCat

The JavaScript Application Server

Product Technical Overview

(02/08/2017)

1. Description

An application server is a software framework that provides both facilities to create Web applications and a server environment to run them[1].

GlassCat is:

- a JavaScript Application Server
- built over NodeJS *(Google)*
- written in TypeScript *(Microsoft)*

GlassCat as:

- a standard for JavaScript Web apps
- a portable development environment
- an efficient alternative to Java EE
- the best integrated solution for modern Web development

[1] Wikipedia: Application server

2. Positioning

GlassCat is an alternative to Java EE server technologies and improves NodeJS/Express[1] development.

Benefits vs. Java EE:

- non-blocking architecture
- stateless native support for REST APIs
- uniform programmatic language
- short learning curve

Benefits vs. NodeJS/Express:

- native full-stack[2] integration
- development standardization
- portability / reusability
- built-in security support
- "Entreprise JavaScript" integration[3]

Lower infrastructure costs
+
Better productivity

[1] Express.js (IBM) [2] back-end + front-end [3] EcmaScript 6 + TypeScript

3. Market Targets

Every company that makes business with the Web applications ecosystem[1].

Needs:

- business-oriented architecture to reduce development costs
- filling the gap between old and modern Web approaches
- infrastructure costs reduction
- stability and "long time" prospect vs. modern Web ecosystems volatility

Segments:

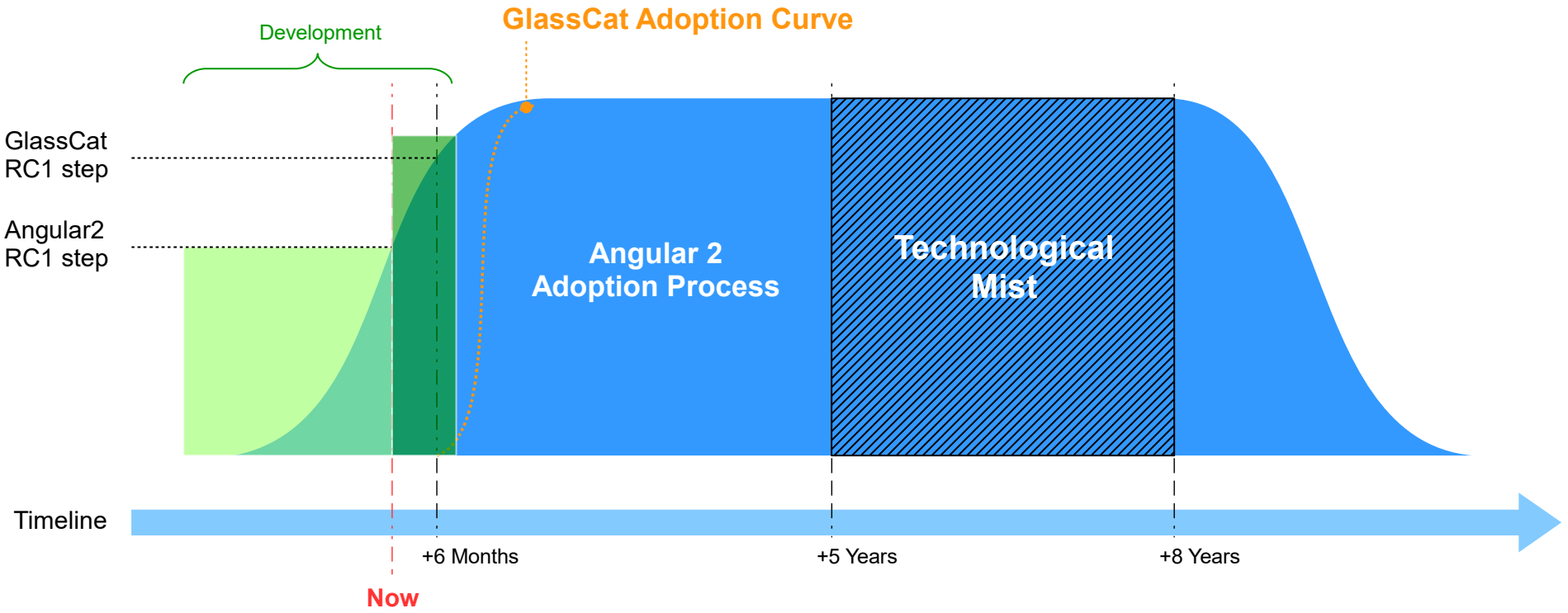
- secured transactions (e.g. banking area)
- Big Data, cloud computing
- REST business services
- Single Page Applications (SPA) with Google Angular
- networking for heavy workloads
- Real Time Communication

[1] Java EE

4. Market Maturity

Since september 2016, Angular2 is the new Google standard for creating SPAs. It is built over the same JavaScript specifications than GlassCat.

Angular 2 is natively integrated to GlassCat ; this is a powerful leverage point for a massive adoption of our product:

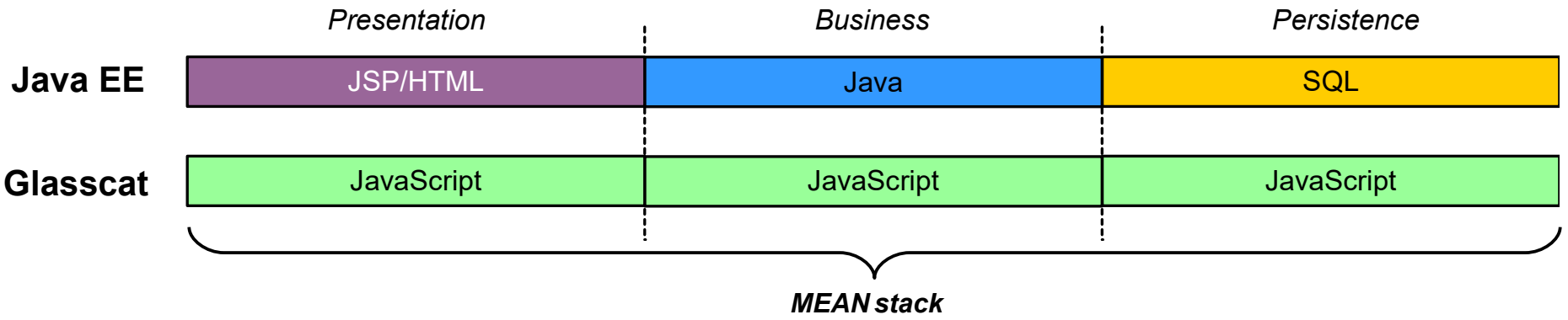


5. The Unified API

Unified APIs offers strong benefits:

- better maintainability
- lower learning curve cost
- better agility integration
- better information system governance

GlassCat uses JavaScript (MEAN[1]) only to manage Web applications layers:



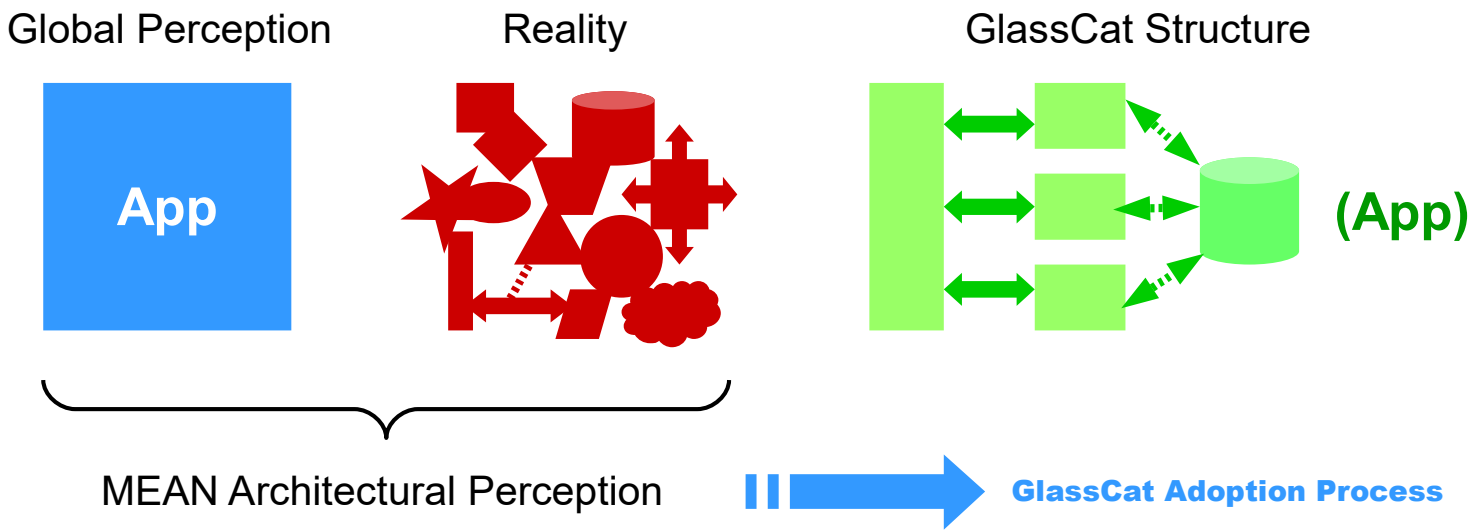
Glasscat provides major improvements to the MEAN[1] stack:

- standardized architecture
- built-in security layer
- built-in development tools
- centralized integration
- packaging support for portability[2]

[1] MongoDB + Express + Angular + Node [2] Similar to Java EE

6. MEAN Perception Issue

Target companies have a wrong perception of the MEAN stack:



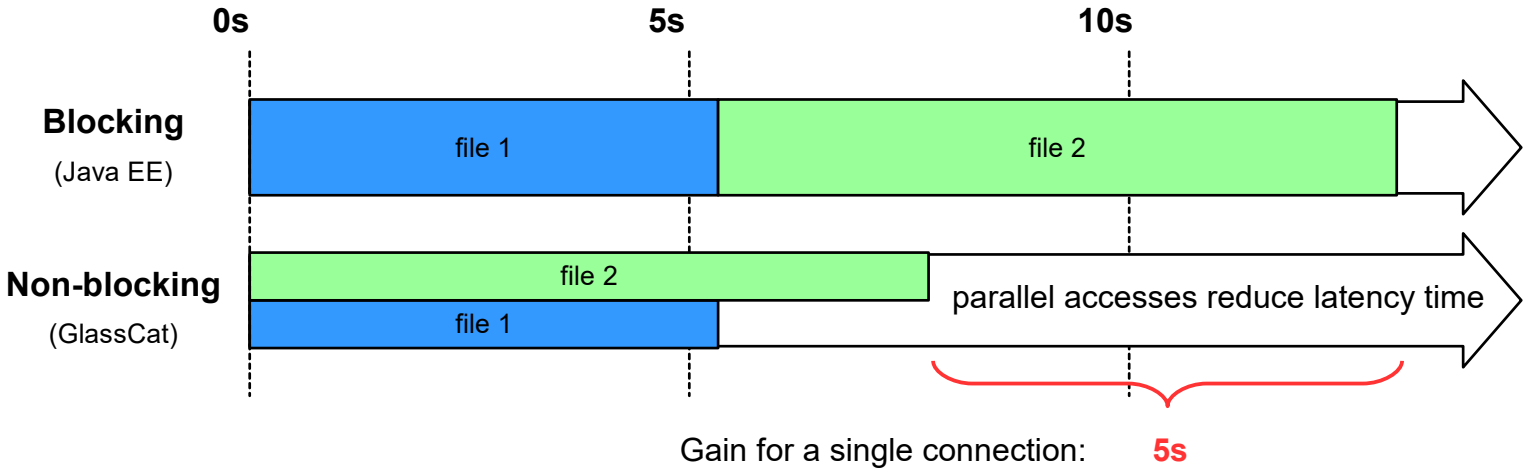
GlassCat provides business-oriented development structure beyond fad:

- faster products development
- better maintainability
- long-term vision

Strong
Costs Reduction

7. Non-blocking Architecture

Non-blocking architectures are expected by the segment targets. They reduce infrastructure costs by mitigating performance issues due to network latency.



Glasscat enables support for tens of thousands of concurrent connections without incurring the cost of thread context switching[1].

[1] Wikipedia : Node.js

8. URL-mapping Architecture

GlassCat URL-mapping (Jslet) is based upon Servlet 3.0 specification. Jslets have the following advantages over other server extension mechanisms:

- they are generally much faster than other server-side scripts[1] because a different process model is used
- they use a standard API
- they have all the advantages of the JavaScript programming language (ease of development + Typescript strong-typing)
- they provide portability and platform independence
- they can access the large set of APIs available for the NodeJS platform, including thousands of projects available from the Node Package Manager (NPM)

GlassCat Jslets are deployed in both, stateless and stateful containers.

GlassCat stateless features provides faster process accesses than Java EE stateful containers.

[1] e.g. CGI...

9. TypeScript Integration

Microsoft TypeScript is a typed superset of JavaScript that compiles to plain JavaScript code which runs in Node.js for:

- class and module support
- static type-checking
- ES6 feature support

GlassCat is entirely written in TypeScript and supports generic types and annotations:

Jslet sample

```
@WebJslet({
  name: "HelloWorldSvc",
  urlPatterns: ["/say-hello"]
})
export class HelloWorldSvc extends HttpJslet {

  public doGet(req:HttpRequest, res:HttpResponse, exit:Function):void {
    let message:string = this.builder.sayHello();
    exit(req, res.send(message));
  }

  @Inject()
  private builder:MessageBuilder;
}
```

10. Productivity

Glasscat provides a huge set of built-in tools for increasing developers productivity.

For example, the following command line:

```
glasscat create-jslet --projectPath=FooBar --name=HelloWorld
```

will produce the Jslet bellow:

```
import {HttpJslet} from "../../../../server/com/onsoft/glasscat/jslets/HttpJslet";
import {WebJslet} from "../../../../server/com/onsoft/glasscat/jslets/WebJslet";
import {HttpRequest} from "../../../../server/com/onsoft/glasscat/net/http/HttpRequest";
import {HttpResponse} from "../../../../server/com/onsoft/glasscat/net/http/HttpResponse";
import {HttpStatusCode} from "../../../../server/com/onsoft/glasscat//net/http/HttpStatusCode";

@WebJslet({
  name: "HelloWorld",
  urlPatterns: ["/hello"]
})
export class HelloWorld extends HttpJslet {

  public doGet(req:HttpRequest, res:HttpResponse, exit:Function):void {
    // TODO Auto-generated method stub
    exit(req, res.send("Hello World!"), null);
  }
}
```

11. Dependency Injection (DI)

Contrary to the MEAN stack GlassCat will include a Spring-based[1] DI mechanism.

GlassCat DI will provide developers native implementation of Inversion of Control (IoC), with the following benefits[2]:

- reduced dependencies
- reduced dependency carrying
- more reusable code
- more testable code
- more readable code

GlassCat DI will support both, annotation and JSON, configuration mechanisms.

[1] https://spring.io/ [2] Dependency Injection Benefits

12. Dependency Management

GlassCat only needs the integrated **Node Package Manager** (NPM) for managing projects dependencies.

GlassCat project structure has been designed to place dependency management of back-side and front-side at the same level.

Additional package managers[1] can be used, to manage front-side applications, without altering the GlassCat dependency management.

Glasscat includes an archetypes management system, to let developers deploy their own Web application structures in one click.

[1] Bower, Webpack...

13. Additional Features

GlassCat includes the following MEAN features:

- Helmet[1] security integration
- server-side customizable templating (similar to JSPs)

GlassCat comes with many non-MEAN additional features:

- additional custom server modules
- built-in timer services
- Java-like security support[2]
- built-in Data Access API
- integrated session mechanisms[2]
- administration console
- projects import/export mechanism

[1] HelmetJS [2] Stateful mode only

14. Micro-project Architecture

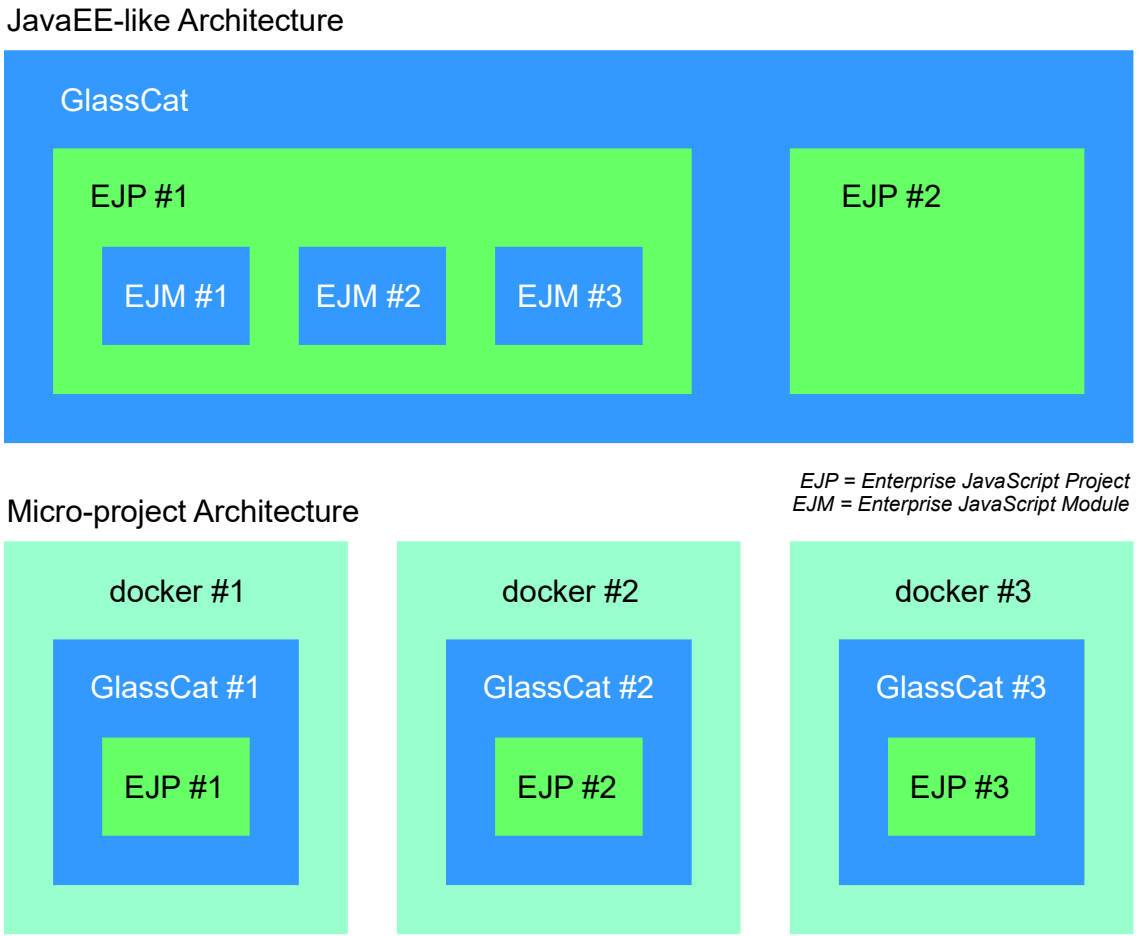
GlassCat supports both, JavaEE-like and Micro-project architectures:

Java EE architectures are:

- team-oriented
- centralized
- less interdependent
- well known

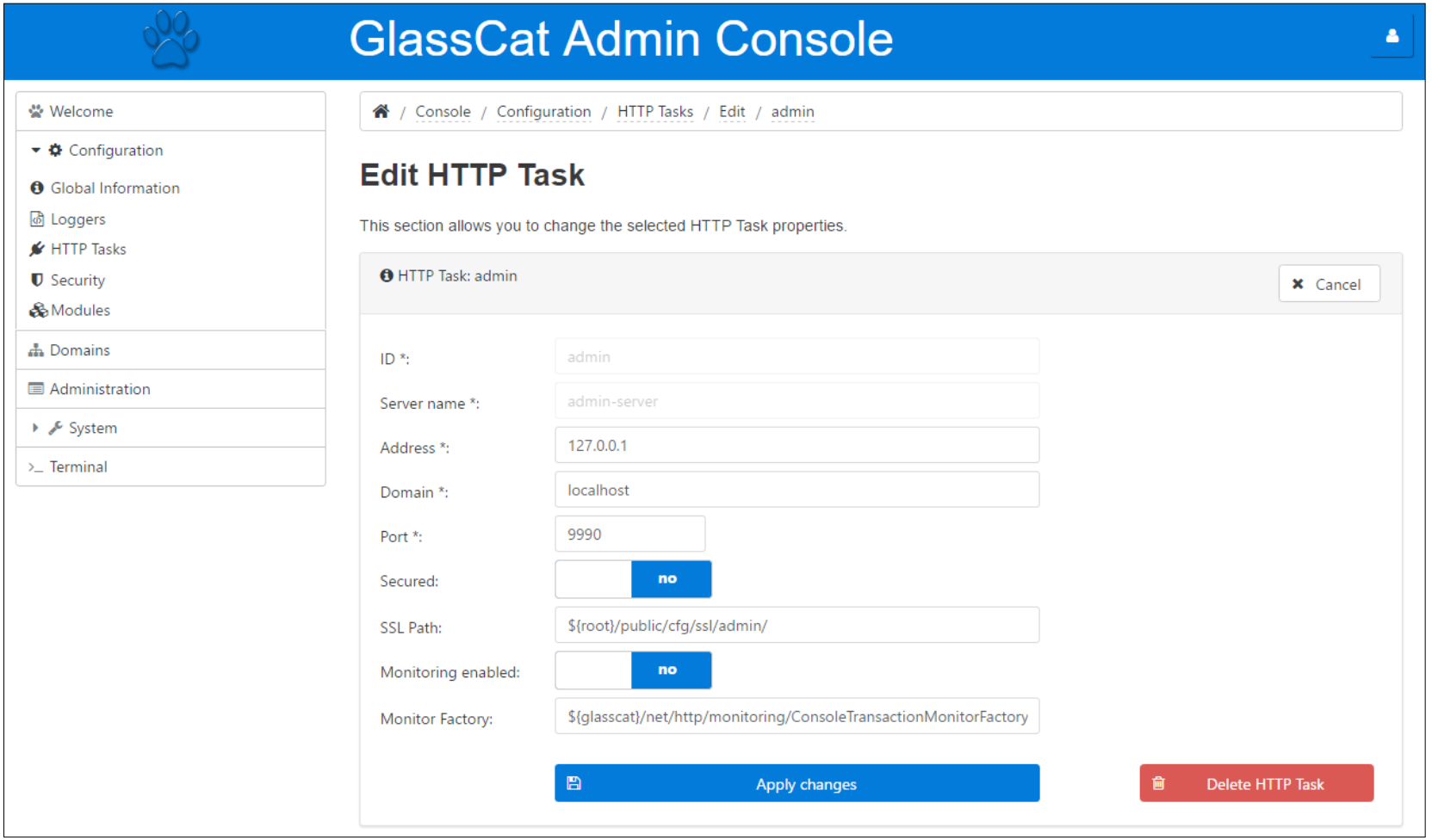
Micro-project architectures are:

- agiles
- developer-oriented
- more maintainable
- easy to deploy



[*] https://www.docker.com/

15. Administration Console



Administration console is built over GlassCat and Angular 2.

16. Development Tools

GlassCat server is built over the JavaScript ecosystem and integrates high-quality open-source technologies provided by major companies in this industry:

	Open-source Tool	Provider
Core platform	NodeJS	Google
Compiler	TypeScript	Microsoft
Editor	Visual Studio Code	Microsoft
HTTP server wrapper	Express 4	Strongloop (IBM)
Apps develoment platform	Angular 2	Google
Dependencies manager	npm	npm, Inc.
Presentation	Bootstrap 4	Twitter
Web components	PrimeNG	PrimeTek[1]

[1] http://www.primefaces.org/whouses.html

17. Roadmap

Release plan:

- RC1: 6 months
- first release: +1 month

Budget plan:

- developers: 2 people
- evangelism: +1 person

	Status	Comments
Core functionalities	100%	complete
Security layer	100%	complete
Administration tools	60%	in progress
Data access API	✗	based on Mongoose
Additional functionalities	✗	filters, timers...
DI, IoC (Spring-like[1])	✗	based on Twister[2]
Testing	✗	based on Jasmine[3]
Documentation	✗	---

[1] https://spring.io/ [2] http://wooz.io/ [3] https://jasmine.github.io/

GlassCat Creator

Technical expert on the Adobe Flash Platform, Pascal is a self-taught developer who focuses on implementing innovative solutions for building Rich Internet Applications.

He has created a Flex-like ActionScript GUI Framework (SPAS 3.0) and a Web Operating System (ONYX OS), with the aim to mix social networking, Web marketing and personal data managment.

His latest project (http://wooz.io) is a powerful JavaScript apps builder, for deploying real softwares in the cloud.

He wrote different XML language specifications to facilitate data exchanges over the Internet, such like Press Releases, or Curriculum Vitæ...

Email: pechemann@gmail.com
Linkedin: www.linkedin.com/in/echemann
Place of residence: French Riviera, GMT+1