



Travel CI

Documentation Projet



Cahier des charges	3
Fonctionnalités attendues	3
Interface utilisateur	3
Conteneurisation	3
WebHooks	4
Journalisation	4
Fonctionnalités facultatives	4
Planification	5
Notification	5
Conclusion	5
Travel CI	6
L'interface Web	6
Liste de projets	6
Ajout / Edition de projet	7
Lancement manuel de Build	8
Liste des builds pour un projet	9
Liste des étapes d'un build	9
Le Back-End	10
Contexte applicatif	10
Détail du système	11
Architecture Spring Cloud	12
Cas d'utilisation : Push sur GitHub	12
Carnet d'échanges	13
Réunions	13
Outils	13
Avancement du projet	14
Elaboration du Product Backlog	14
13/04 Structure du projet et présentation du template	15
27/04 Gestion des outils de versionning	16
18/05 Gestion des projets et des commandes à exécuter	17
08/06 Lancement du build d'un projet	18
29/06 Gestion des logs	19

Cahier des charges

Vous trouverez ci-dessous la retranscription du cahier des charges du projet tel qu'il nous a été fourni par nos clients.

Fonctionnalités attendues

Interface utilisateur

De manière à permettre une interaction aisée avec la solution produite, il est attendu qu'une **interface web** permette la configuration des projets d'intégration continue. Il n'est pas nécessaire de mettre en place une authentification ; nos proxy sont en charge des accès aux différents serveurs présents sur notre réseau local.

Cette interface utilisateur permettra la **gestion complète des projets** : création, mise à jour, suppression et consultation.

Pour chaque projet, il sera nécessaire de pouvoir ajouter un nombre indéfini de **commandes** qui constitueront le processus d'intégration de l'application.

Il sera bien entendu nécessaire de pouvoir consulter les logs d'exécution de chacune des commandes lors de son exécution, et de conserver ces données de journalisation pour un possible usage ultérieur.

Conteneurisation

L'exécution d'une tâche de CI devra être effectuée dans un **conteneur Docker**, ce qui permettra de conserver une grande flexibilité dans nos futurs projets sans nous lier à une technologie en particulier.

Ainsi, pour chaque projet, il sera nécessaire de fournir un **Dockerfile** qui servira de base au conteneur d'intégration. Lors de l'exécution du CI, un conteneur sera créé à partir de cette image, un volume contenant les sources du projet sera partagé sur ce conteneur, et les différentes commandes configurées depuis l'interface web (voir *Interface utilisateur*) seront exécutées dans le conteneur les unes à la suite des autres. Si une commande échoue, l'ensemble de la tâche de CI est interrompue. Il est nécessaire de pouvoir connaître l'état d'exécution de chacune des commandes.

Concernant le déploiement continu, nous utiliserons les **CLI** des différents fournisseurs de solutions d'hébergement Cloud (Microsoft Azure, Amazon Web Services...) directement dans les commandes configurées dans un projet pour déployer l'exécutable si l'ensemble des commandes précédentes se sont correctement exécutées.

WebHooks

Nos équipes de développement utilisent une solution Atlassian BitBucket hébergée en interne pour la gestion des dépôts Git des projets. Travel CI devra pouvoir être connecté aux **webhooks** fournis par BitBucket de manière à être averti lorsque du code est mis en ligne sur nos dépôts pour qu'il puisse le récupérer et déclencher une exécution du CI.

Journalisation

Comme expliqué précédemment, nous souhaitons que vous apportiez un soin particulier à la journalisation, car il s'agit de la clé pour identifier la provenance d'une erreur en cas d'échec d'une tâche d'intégration continue.

Nous souhaitons donc que vous mettiez en place une solution centralisant la gestion de la journalisation pour l'ensemble du système. Encore une fois, il sera nécessaire que nous puissions visualiser ces données depuis l'interface web de gestion.

Fonctionnalités facultatives

Les fonctionnalités suivantes sont considérées comme étant **facultatives**, ce qui signifie qu'elles ne sont pas attendues dans le livrable final Travel CI.

S'il advient cependant que le développement de la solution prenne de l'avance ou que ces fonctionnalités intéressent particulièrement vos équipes de développement, nous sommes ouverts à la discussion pour les intégrer dans un potentiel livrable supplémentaire.

Planification

Outre la possibilité d'exécuter les processus d'intégration continue suite à la réception d'un webhook, il nous a semblé intéressant de pouvoir **planifier** l'exécution de ces processus.

Un cas d'application concret serait de planifier une tâche de déploiement en production en dehors des périodes d'activités sur nos différentes plateformes.

Notification

De manière à améliorer la réactivité de nos équipes de développement, la mise en place d'un système de notification de l'état d'un processus d'intégration continue à l'issue de son exécution pourrait être un véritable plus.

Les deux systèmes de communication les plus utilisés par nos équipes sont les **e-mails** ainsi que l'application de messagerie instantanée **Slack**.

Il s'agit encore une fois de fonctionnalités facultatives dont l'intégration n'est pas attendue dans le livrable final.

Conclusion

Gardez à l'esprit que notre intérêt *commun* est que vous produisiez une solution de la meilleure qualité possible.

Dans ce sens, nous nous efforçons d'être toujours disponibles et à votre écoute en ce qui concerne l'avancement du projet, les différentes fonctionnalités ou approches métiers souhaitées, etc...

N'hésitez pas à vous montrer moteurs et à prendre des initiatives fonctionnelles comme techniques, car un bon projet est un projet que l'on prend plaisir à faire évoluer !

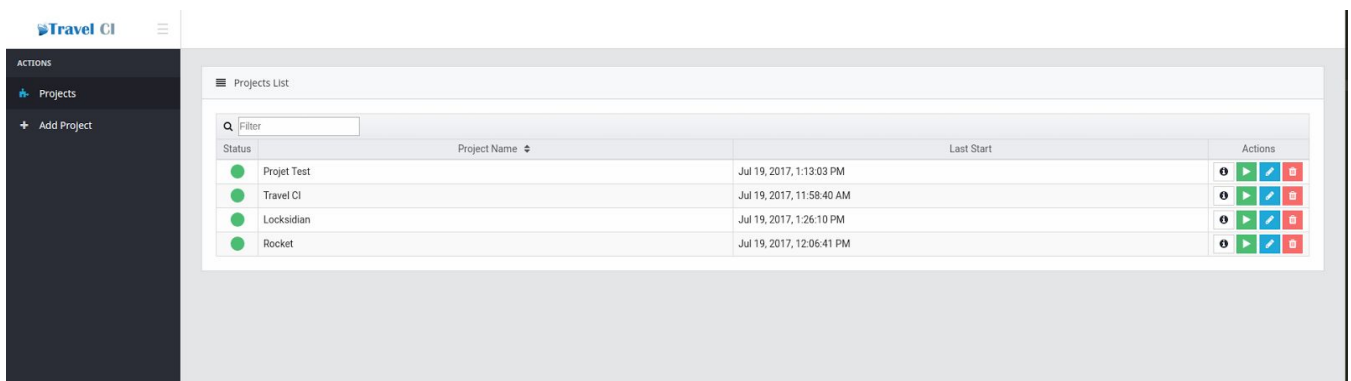
Travel CI

La solution Travel CI est décomposée en 2 parties : Une application Front-End développée en Angular 4 permettant la saisie d'informations et la visualisation de l'état de chacun des projet gérés et un ensemble d'applications microservices formant ainsi la partie Back-End.

L'interface Web

Liste de projets

Premier écran visible lors du lancement de l'application Web, la liste des projets permet de visualiser les projets gérés par la solution Travel CI et obtenir des informations sur leurs états, dernier lancement, ...



The screenshot shows the 'Travel CI' web interface. On the left is a dark sidebar with 'ACTIONS' and 'Projects' (with a plus icon and 'Add Project' link). The main area is titled 'Projects List' and contains a table with a search filter. The table has columns for Status, Project Name, Last Start, and Actions. It lists four projects: 'Projet Test', 'Travel CI', 'Locksidian', and 'Rocket', all with green status indicators and timestamps from July 19, 2017. Each project has a set of action buttons (play, edit, delete).

Status	Project Name	Last Start	Actions
●	Projet Test	Jul 19, 2017, 1:13:03 PM	[play] [edit] [delete]
●	Travel CI	Jul 19, 2017, 11:58:40 AM	[play] [edit] [delete]
●	Locksidian	Jul 19, 2017, 1:26:10 PM	[play] [edit] [delete]
●	Rocket	Jul 19, 2017, 12:06:41 PM	[play] [edit] [delete]

A partir de cette interface, il est également possible de lancer manuellement un build, obtenir l'historique des builds, éditer un projet ou encore le supprimer.

Ajout / Edition de projet

Il est possible d'ajouter ou d'éditer les informations d'un projet. Cette interface permet de saisir les informations nécessaires au fonctionnement de la solution (Url de repository, emplacement d'un Dockerfile, commandes à exécuter, ...)

Travel CI

ACTIONS

Projects

Add Project

Edit Project

Name

Travel CI

Description

Travel CI build by Travel CI

Enable

YES

Repository Url

https://github.com/Travel-CI/tci-project

Branches

master, dev

Dockerfile Location

/travelci/

Edit Project

Cancel

Commands

Order	Command	Enable	Logs	Action
1	ls -al	YES	YES	
2	chmod +x -R .	YES	NO	
3	sed -i 's/import org.jl	YES	NO	
4	sed -i 's/@Test/@Tee	YES	NO	
5	./gradlew clean jacoc	YES	YES	
6	cd tci-webapp && npr	YES	YES	
7	cd tci-webapp && ng	YES	YES	

Add

Powered by TCI-Team

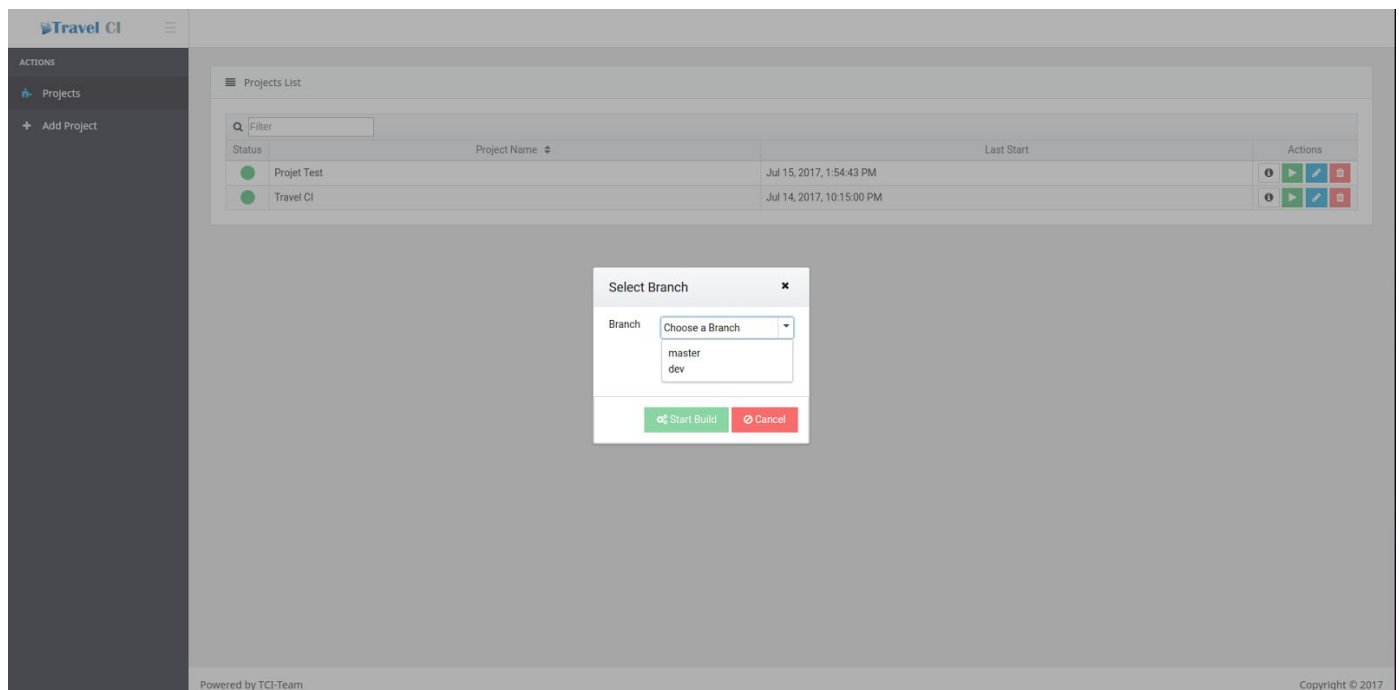
Copyright © 2017

Lancement manuel de Build

Travel CI est une solution démarrée par deux évènements :

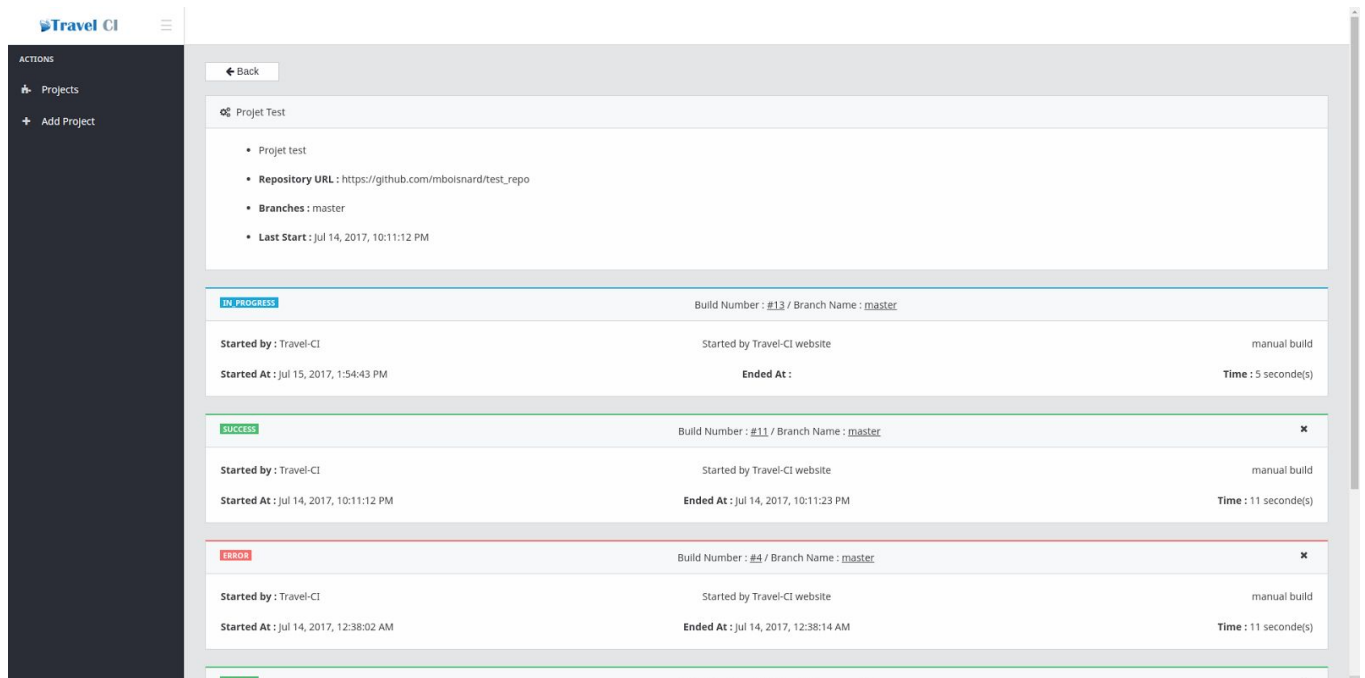
- La modification de code enregistrée dans un système de versionning (Bitbucket / GitHub)
- Le lancement manuel effectué par l'utilisateur

Avant de lancer le build manuellement, l'utilisateur sera interrogé sur le choix de la branche GIT qui sera prise en compte par le système.



Liste des builds pour un projet

Un écran récapitulatif des builds effectués est disponible pour chaque projet. On y retrouve le statut du build (success, in progress, error) ainsi que certaines informations (heure de lancement, heure de fin, message de commit, ...).

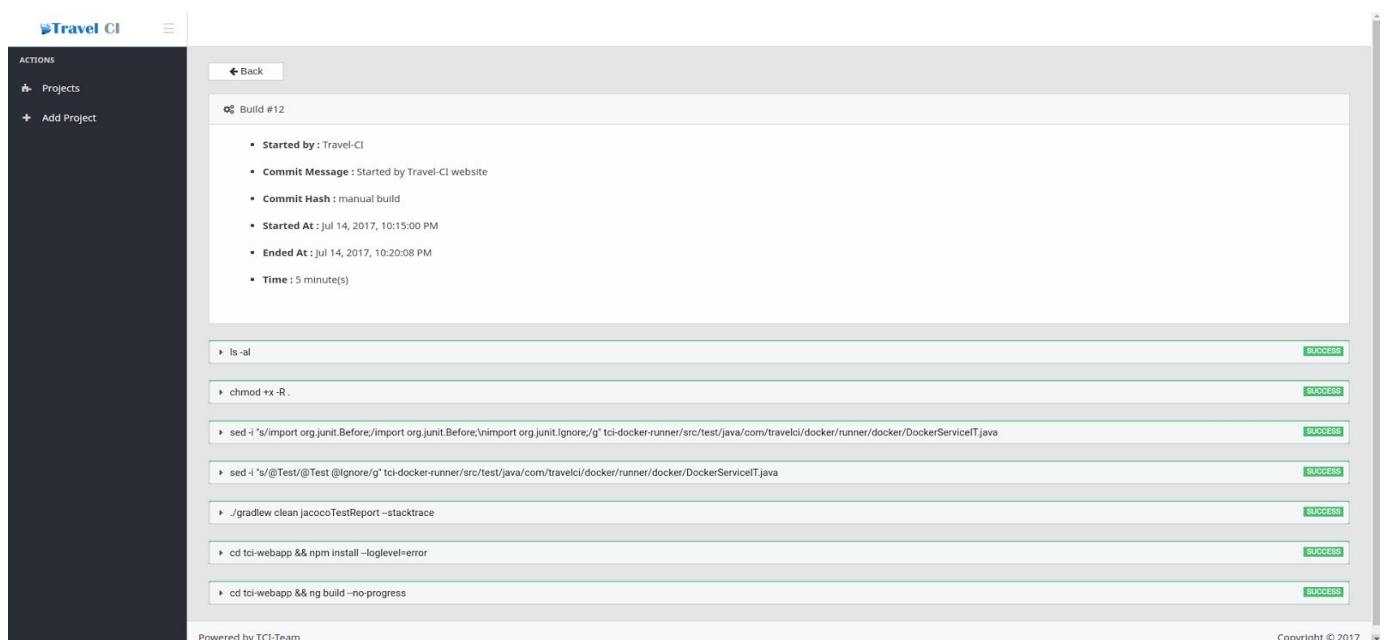


The screenshot shows the Travel CI interface. On the left is a sidebar with 'ACTIONS' and 'Projects'. The main area displays a 'Project Test' summary with details like Repository URL and Last Start. Below this is a table of builds:

Build Number	Branch Name	Status	Started by	Started At	Ended At	Time
#13	master	IN PROGRESS	Travel-CI	Jul 15, 2017, 1:54:43 PM		5 seconde(s)
#11	master	SUCCESS	Travel-CI	Jul 14, 2017, 10:11:12 PM	Jul 14, 2017, 10:11:23 PM	11 seconde(s)
#4	master	ERROR	Travel-CI	Jul 14, 2017, 12:38:02 AM	Jul 14, 2017, 12:38:14 AM	11 seconde(s)

Liste des étapes d'un build

Enfin, nous retrouvons sur cette interface, les différentes commandes exécutées lors d'un build, ainsi que leurs résultat d'exécution.



The screenshot shows the Travel CI interface for 'Build #12'. It displays a summary of the build and a list of executed commands with their results:

- Started by: Travel-CI
- Commit Message: Started by Travel-CI website
- Commit Hash: manual build
- Started At: Jul 14, 2017, 10:15:00 PM
- Ended At: Jul 14, 2017, 10:20:08 PM
- Time: 5 minute(s)

Command	Result
ls -al	SUCCESS
chmod +x -R	SUCCESS
sed -i 's/import org.junit.Before/import org.junit.Before;\\nimport org.junit.Ignore;\\n' tci-docker-runner/src/test/java/com/travelci/docker/runner/docker/DockerServiceIT.java	SUCCESS
sed -i 's/@Test/@Test @Ignore;\\n' tci-docker-runner/src/test/java/com/travelci/docker/runner/docker/DockerServiceIT.java	SUCCESS
./gradlew clean jacocoTestReport --stacktrace	SUCCESS
cd tci-webapp && npm install --loglevel=error	SUCCESS
cd tci-webapp && ng build --no-progress	SUCCESS

Powered by TCI-Team Copyright © 2017

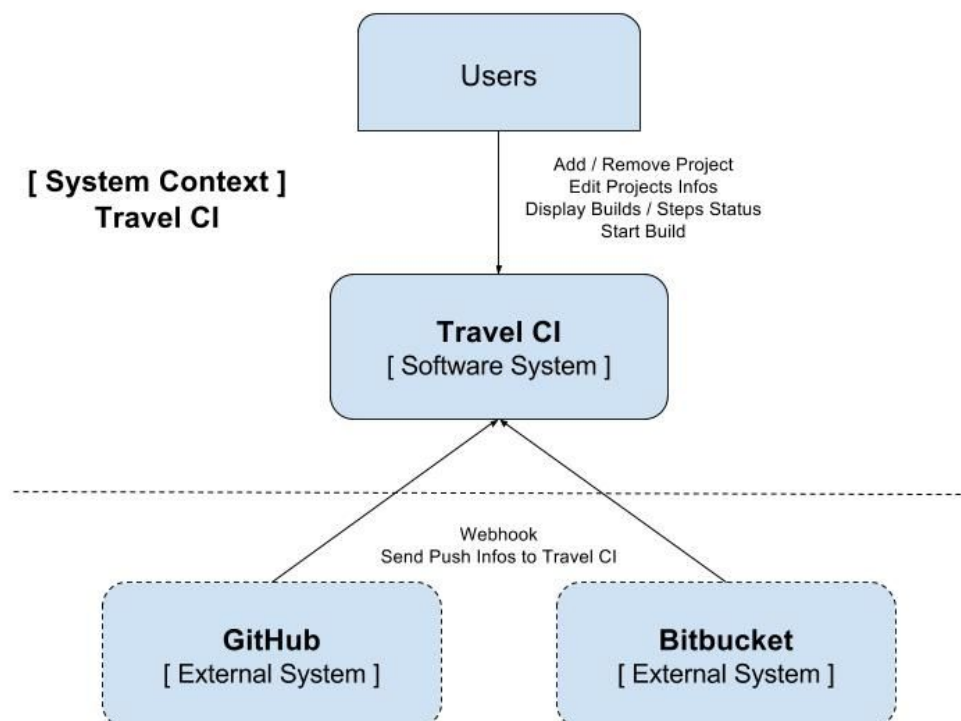
Le Back-End

La partie Back-End de Travel CI est développée en Java et utilise le framework Spring Cloud. Ce dernier offre de nombreuses possibilités et permet le découpage en microservices. Il est ainsi possible d'envisager la scalabilité de chacune des fonctionnalités du projet en fonction de la charge à un instant T.

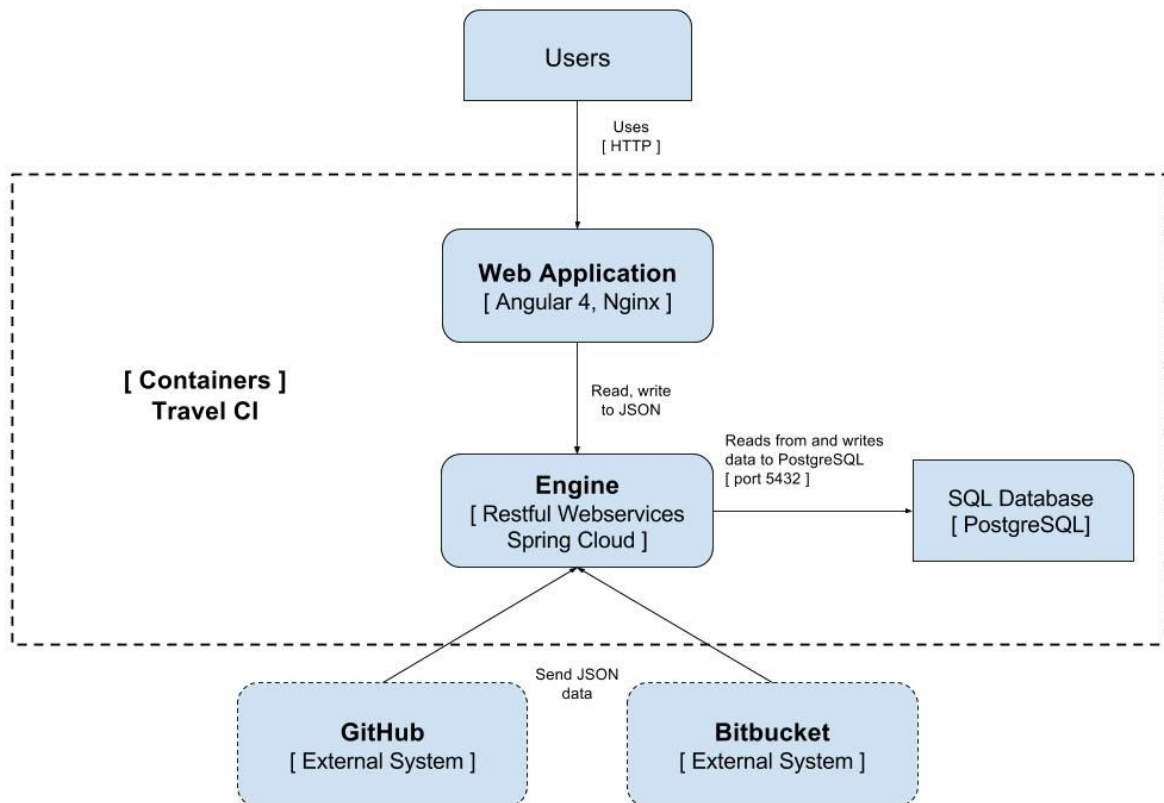
De plus, afin d'isoler totalement un projet en cours de build du reste de la solution et de permettre une grande flexibilité d'utilisation, nous exploitons la technologie Docker. Chaque projet doit disposer d'un fichier descriptif, appelé Dockerfile, indiquant l'environnement nécessaire afin pouvoir effectuer des actions sur le projet.

Vous trouverez ci-dessous les différents diagrammes expliquant le fonctionnement en microservices de notre solution.

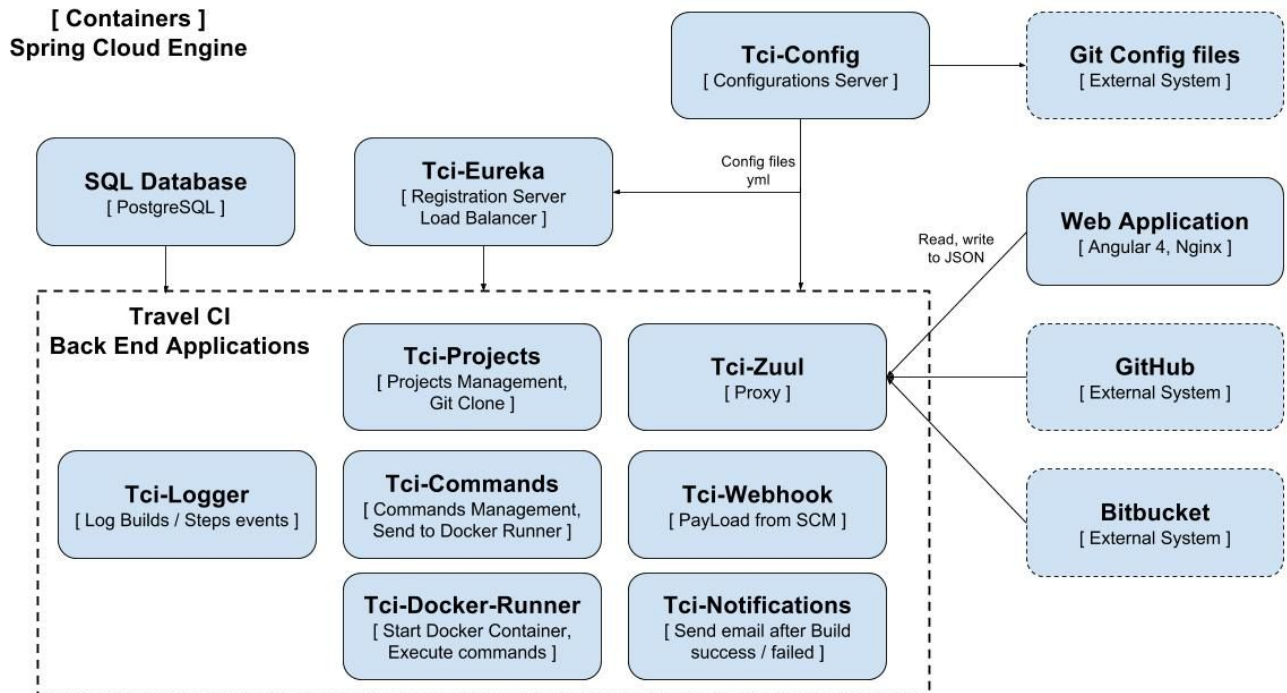
Contexte applicatif



Détail du système

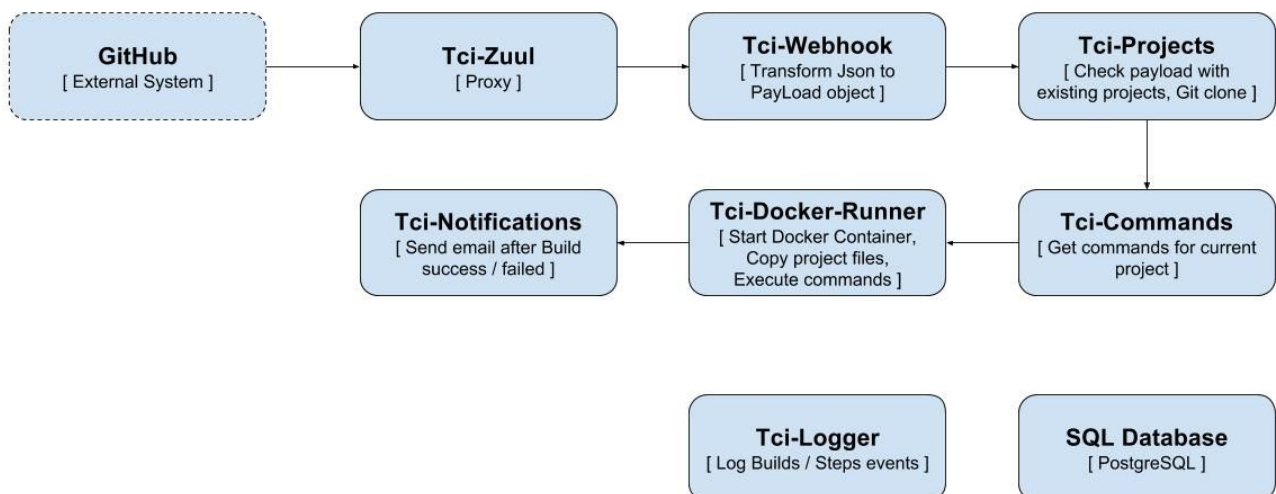


Architecture Spring Cloud



Cas d'utilisation : Push sur GitHub

Use Case Push on GitHub



Carnet d'échanges

Réunions

Afin de maximiser les retours utilisateurs et ainsi être à l'écoute des changements à apporter, nous avons choisi de démarrer ce projet en utilisant une méthodologie Agile. Avec l'accord de notre client, nous avons utilisé la méthode SCRUM et avons pu planifier des itérations et des rendez-vous pendant lesquels nous avons discuté de l'avancement du projet et des possibles modifications à apporter. Chaque itération, sprint, a une durée de **3 semaines**. De cette façon, nos échanges client-prestataire ont pu s'effectuer tous les jeudi après-midi des semaines de cours. Lors de chacune des nos réunions, nous avons pris en note les remarques de notre client et les avons ensuite reportés dans nos outils internes.

Outils

Une solution de collaboration et d'historisation des demandes du client nous a été nécessaire afin de mener à bien ce projet. En effet, notre équipe composée de 3 personnes ne travaille pas dans les mêmes locaux et a donc besoin d'une plateforme afin de visualiser l'avancement du projet ainsi que les tâches restants à effectuer. Nous avons utilisé la solution **Atlassian Jira** pour gérer les différents scénarios utilisateurs (*user story*) ainsi les tâches et les bugs à corriger. Cette solution est également couplée au logiciel **Atlassian Bitbucket** afin d'avoir sur le code correspondant à chacune des demandes.

Avancement du projet

Elaboration du Product Backlog

Lors de la première réunion avec le client, nous avons repris l'intégralité du cahier de charges et avons créé le Product Backlog correspondant.


Backlog 9 demandes

		TCI-33 Ajout / Edition / Suppression d'un projet depuis l'interface web
		TCI-34 Lancement des builds depuis l'interface web
		TCI-35 Visualisation des logs depuis l'interface web
		TCI-36 Gestion des étapes de build depuis l'interface web
		TCI-37 Outils de versionning démarre le build automatiquement
		TCI-16 Technique : Découpage Microservices
		TCI-12 Technique : Structure du projet
		TCI-38 Utilisation de Docker pour builder les projets
		TCI-39 Technique : Supporter la charge avec Spring Cloud

De plus, nous avons pu fixer les différentes dates de rendez-vous afin que nous puissions obtenir un retour sur le développement du projet.

13/04 Structure du projet et présentation du template

Lors de ce premier sprint de 3 semaines, nous nous sommes focalisés sur la compréhension du fonctionnement des technologies que nous allons utiliser pour ce projet et nous avons présenté un premier aperçu visuel de l'interface web. Les retours du client sur cette dernière ont plutôt été satisfaisant, car elle était intuitive et simple à comprendre.

 Travel CI / TCI-12

Technique : Structure du projet

✎ Modifier

💬 Commentaire

Attribuer

Suite ▾

A faire

En cours

Fin

Gérer ▾


Informations

Type:	■ Récit	Etat:	FINI (Afficher le flux de travaux)
Priorité:	↑ High	Résolution:	Terminé
Étiquettes:	Aucune		
Sprint:	Étude et Découverte, Evolution de la solution		








Description

Préparation de la structure du projet pour livrer rapidement des fonctionnalités.
Etude du fonctionnement des nouvelles technologies utilisées dans le projet (Spring Cloud, Spring Boot, Angular).
Recherches et POC

Pièces jointes


 Glissez-déposez des fichiers pour les joindre, ou [parcourir](#).

Sous-tâches


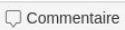






1. ✓ Recherche architecture microservice	 FINI Mathieu Boisdard
2. ✓ Création Structure Spring Boot	 FINI Mathieu Boisdard
3. ✓ Environnement Docker	 FINI Mathieu Boisdard
4. ✓ Fonctionnement d'Angular	 FINI Julien Bertauld
5. ✓ Template Angular	 FINI Jérémy Boehm
6. ✓ Intégration d'un template HTML	 FINI Jérémy Boehm
7. ✓ Exploration de Spring Boot	 FINI Julien Bertauld

27/04 Gestion des outils de versionning




Le déclenchement automatique du build d'un projet est une fonctionnalité essentielle dans la solution Travel CI, il était donc logique de commencer par cette dernière. La démonstration lors de la réunion fut donc de montrer que notre système interagissait bien avec les 2 acteurs majeurs du versionning de code (Bitbucket et Github).

 Travel CI / TCI-37

Outils de versionning démarre le build automatiquement


Informations

Type:	 Récit	Etat:	 (Afficher le flux de travaux)
Priorité:	 High	Résolution:	Terminé
Étiquettes:	Aucune		
Sprint:	Evolution de la solution		







Description


Un outil de versionning (Bitbucket, Github) peut démarrer le build d'un projet par l'intermédiaire d'un WebHook

Pièces jointes


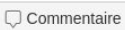








Sous-tâches




1.  Fonctionnement du webhook	 Julien Bertauld
2.  Création méthode d'analyse des données Bitbucket / Github	 Julien Bertauld
3.  Tests unitaires et d'intégration	 Julien Bertauld

 Travel CI / TCI-16

Technique : Découpage Microservices


Informations

Type:	 Récit	Etat:	 (Afficher le flux de travaux)
Priorité:	 Medium	Résolution:	Terminé
Étiquettes:	Aucune		
Sprint:	Evolution de la solution		









Description

Structurer les différents microservices pour faciliter l'intégration des fonctionnalités

Pièces jointes




Sous-tâches

1.  Microservice Projets	 Jérémy Boehm
2.  Microservice Commands	 Jérémy Boehm
3.  Microservice DockerRunner	 Mathieu Boisnard
4.  Microservices Spring Cloud	 Mathieu Boisnard

18/05 Gestion des projets et des commandes à exécuter

Ce sprint fut dédié à l'interface web et aux interactions qu'elle peut avoir avec notre système Back-End.

Travel CI / TCI-33

Ajout / Edition / Suppression d'un projet depuis l'interface web

ModifierCommentaireAttribuerSuiteA faireEn coursFiniGérer

Informations

Type: Récit

Priorité: Medium

Étiquettes: Aucune

Sprint: Evolution de la solution

Etat: FINI (Afficher le flux de travaux)

Résolution: Terminé

Description

Un utilisateur peut ajouter / supprimer / éditer un projet depuis son interface web

Pièces jointes

Glissez-déposez des fichiers pour les joindre, ou parcourir.

Sous-tâches

1. Création Page Liste de projets

FINI

Julien Bertauld

2. Création Page Ajout / Edition d'un projet

FINI

Jérémy Boehm

3. Modification Branches Projects

FINI

Jérémy Boehm

4. Récupérer un projet GIT dans tci-projects

FINI

Mathieu Boisdard

08/06 Lancement du build d'un projet

La fonctionnalité de lancement de build avec l'utilisation d'images Docker fut compliquée à implémenter. Il a donc fallu plusieurs sprints avant qu'une démonstration devant le client soit envisageable.



Travel CI / TCI-36

Gestion des étapes de build depuis l'interface web

[Modifier](#) [Commentaire](#) [Attribuer](#) [Suite](#) [A faire](#) [En cours](#) [Fini](#) [Gérer](#)

Informations

Type: **Récit** Etat: **FINI** (Afficher le flux de travaux)
Priorité: **Medium** Résolution: Terminé
Étiquettes: Aucune
Sprint: Evolution de la solution

Description

Un utilisateur peut ajouter / supprimer / éditer des étapes de build depuis son interface web

Pièces jointes

Glissez-déposez des fichiers pour les joindre, ou [parcourir](#).

Sous-tâches

1. [Ajout lancement de build depuis Angular](#) **FINI** Julien Bertauld
2. [Interaction avec le microservice projet](#) **FINI** Jérémy Boehm



Travel CI / TCI-38

Utilisation de Docker pour builder les projets

[Modifier](#) [Commentaire](#) [Attribuer](#) [Suite](#) [A faire](#) [En cours](#) [Fini](#) [Gérer](#)

Informations

Type: **Récit** Etat: **FINI** (Afficher le flux de travaux)
Priorité: **Medium** Résolution: Terminé
Étiquettes: Aucune
Sprint: Evolution de la solution

Description

L'utilisation de Docker et des conteneurs est nécessaire afin d'assurer une grande flexibilité (build de tous type de projets), et apporte une sécurité plus importante (isolation dans un conteneur, "sandbox")

Pièces jointes


Glissez-déposez des fichiers pour les joindre, ou [parcourir](#).

Sous-tâches

1. [Exécuter des commandes dans le container](#) **FINI** Mathieu Boisnard
2. [Builder une image Docker - Docker-Runner](#) **FINI** Mathieu Boisnard

29/06 Gestion des logs

Après avoir développé la structure et le moteur de la solution Travel CI, il nous restait à gérer les logs afin d'obtenir des informations sur le déroulement des builds des projets, sur les potentielles erreurs lors de l'exécution du système.

 Travel CI / TCI-35

Visualisation des logs depuis l'interface web

Modifier

Commentaire

Attribuer

Suite ▾



A faire

En cours

Fin

Gérer ▾


Informations

Type:	 Récit	Etat:	FINI (Afficher le flux de travaux)
Priorité:	 Medium	Résolution:	Terminé
Étiquettes:	Aucune		
Sprint:	Evolution de la solution		







Description

Un utilisateur peut voir les logs / erreurs des builds depuis son interface web

Pièces jointes

 Glissez-déposez des fichiers pour les joindre, ou [parcourir](#).

Sous-tâches

1.  Ajouter un microservice dédié aux logs	 FINI Mathieu Boisdard
2.  Appeler le service de logs (infos microservices, errors, docker, ...)	 FINI Julien Bertauld
3.  Visualiser les logs sur l'appli Angular	 FINI Jérémy Boehm