

SYSTEME DE RESERVATION EN LIGNE D'UN RESTAURANT D'UN L'HOTEL :

Travail réalisé par :

Nasroallah Saad / Srrhir Sami

CODE SOURCE (GITHUB)

Front end: <https://github.com/samisrrhir/Front-End-Food-Ordering-App-Angular>

Back end: <https://github.com/samisrrhir/microservice-using-Eureka-Spring-boot-Delievry-App>

ARCHITECTURE

Backend (Spring Boot) :

- Utilisation de Spring Boot pour la partie Back-End.
- Structuration en couches (contrôleurs, services, repository) pour séparer les packages.
- Gestion des commandes, des repas disponibles, du panier et des transactions.

Frontend (Angular) :

- Interface utilisateur conviviale pour afficher les repas disponibles, gérer le panier et finaliser la commande.
- Communication avec l'API backend via des appels HTTP pour récupérer les données et soumettre les commandes.
- Dashboard Admin responsable .

OVERVIEW DE L'APPLICATION

Le projet de commande de repas est une application qui permet aux clients de parcourir un menu de restaurant, de sélectionner des repas et de passer des commandes dans un restaurant. Le système doit gérer les commandes, les repas, et le suivi des stocks tout en offrant une expérience fluide et efficace.

Présentation Fonctionnelle :

Client :

Sélection de Repas : L'utilisateur peut parcourir le menu, consulter les détails des repas et ajouter des articles à son panier.

Création de Commande : L'utilisateur peut créer une commande en sélectionnant plusieurs repas et spécifier des préférences si nécessaire.

Gestion de Panier : La possibilité de visualiser, ajuster et supprimer des éléments du panier avant la confirmation de la commande.

Commande :

Liste de Repas: Une commande est constituée d'une liste de repas sélectionnés par l'utilisateur.

Prix Total: Chaque commande affiche le prix total basé sur les repas sélectionnés.

Repas:

Détails du Repas :

Chaque repas a des détails tels que le nom, la description, le prix, etc.

Stocks : Le système doit suivre les stocks de chaque repas pour éviter la commande d'articles épuisés.

Structuration du Code:

Le code est bien structuré avec une séparation claire des responsabilités entre les différentes couches de l'application (contrôleurs, services, repository).

Les noms de classes, méthodes et variables sont descriptifs, facilitant la compréhension du code source.

Gestion des Erreurs:

La gestion des erreurs est mise en place de manière robuste, assurant une expérience utilisateur sans faille.

Les messages d'erreur sont clairs et informatifs, facilitant le débogage et la résolution des problèmes.

Tests Unitaires:

Une couverture significative de tests unitaires est présente, garantissant une validation efficace des fonctionnalités.

Les tests sont écrits de manière à couvrir les cas limites et les scénarios critiques.

Performance:

L'application offre des performances satisfaisantes, avec des temps de réponse rapides lors de l'interaction avec l'interface utilisateur.

Les requêtes vers le backend sont optimisées pour minimiser les temps de latence.

Interface Utilisateur (Frontend):

L'interface utilisateur est conviviale, avec une navigation intuitive à travers le menu et le panier.

Les interactions utilisateur, telles que l'ajout d'articles au panier, sont fluides et réactives.

Fonctionnalités Client:

Les fonctionnalités client, telles que la sélection de repas, la création de commandes et la gestion du panier, sont implémentées de manière intuitive et efficace.

Les préférences utilisateur sont prises en compte de manière appropriée lors de la création des commandes.

Maintenabilité du Code:

Le code est maintenable avec des commentaires appropriés, permettant à d'autres développeurs de comprendre rapidement le fonctionnement des différentes parties de l'application.

Les mises à jour et les modifications peuvent être effectuées de manière sûre et efficace.

Rapports SonarQube:

Les rapports SonarQube soulignent la qualité du code en identifiant les zones nécessitant une attention particulière.

Les corrections recommandées dans les rapports SonarQube sont régulièrement prises en compte pour maintenir des normes élevées de développement.

Ce rapport met en évidence les aspects clés de l'assurance qualité de l'application sans entrer dans les détails spécifiques de l'architecture.

DIAGRAMME DE GANT

Eat IT fast food ordering:				10/Nov	18/Nov		90%	
1	✓ Identification des Exigences	Unassigned	-	10/Nov	18/Nov	Finished	100%	
2	✓ PAQ	Unassigned	-	13/Nov	18/Nov	Finished	100%	
3	✓ Conception UML	Unassigned	-	14/Nov	18/Nov	Finished	100%	
4	✓ Maquettes Interfaces	Unassigned	-	15/Nov	18/Nov	Finished	80%	
5	✓ Back end : Architecture Spring MVC	Unassigned	-			Finished	100%	
6	✓ Developpement de l'interface utilisateur	Unassigned	-			Started	60%	
7	✓ Developpement de l'interface admin	Unassigned	-			Started	30%	
8	✓ tests unitaires	Unassigned	-			Started	80%	
9	✓ Test fonctionnels:Selenium	Unassigned	-			Started	80%	
10	✓ Sonar Qube	Unassigned	-			Started	80%	

DIAGRAMME DE CAS D'UTILISATION

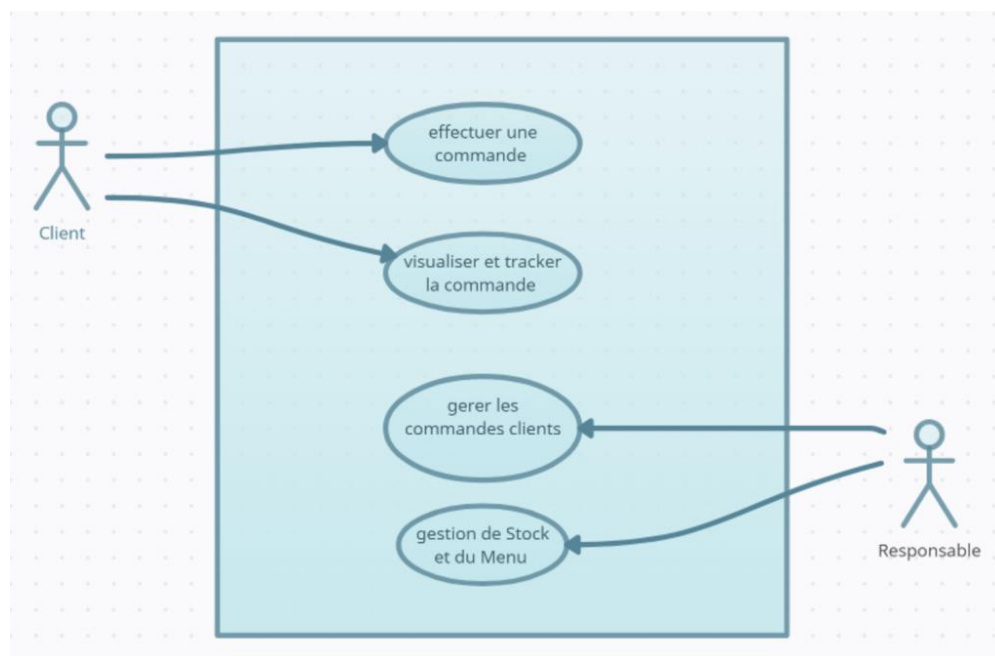
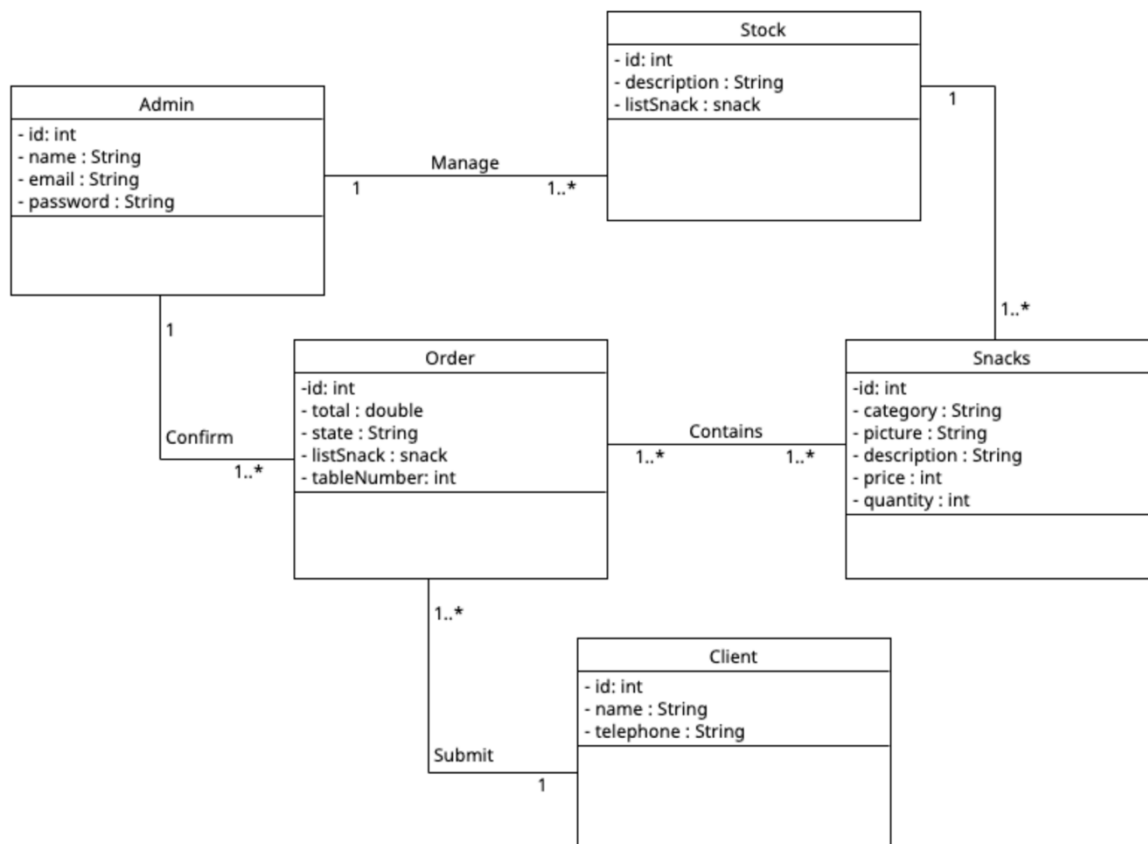
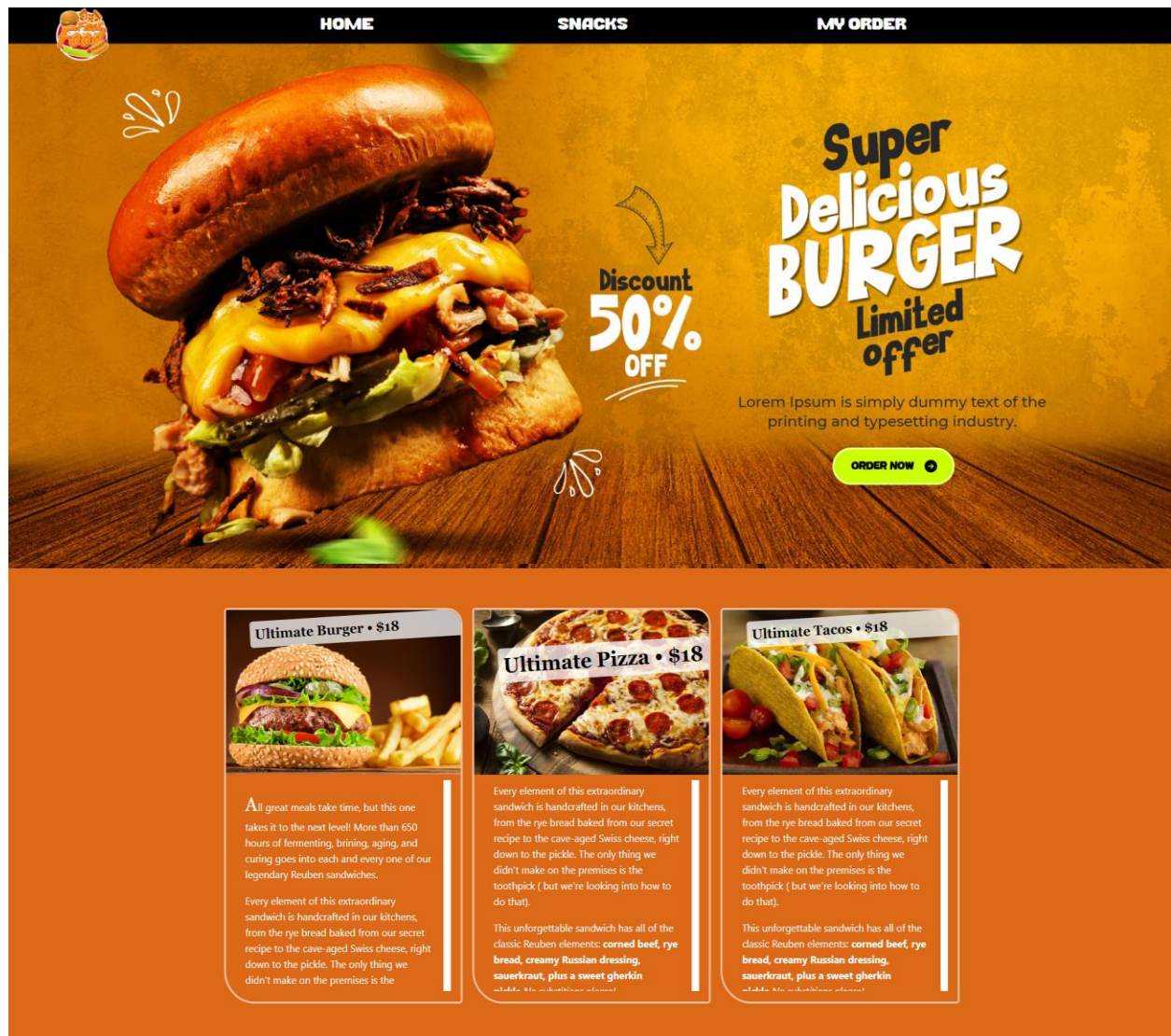



DIAGRAMME DE CLASSE







HOME

SNACKS

MY ORDER

HELLO CUSTOMER !

Hamburger 🍔

Juicy beef patty topped with fresh lettuce, tomatoes, onions, and a special sauce, all nestled in a soft, toasted bun.

Pizza 🍕

A delightful medley of melted cheese, savory tomato sauce, and a variety of delicious toppings on a thin, crispy crust.

Hotdog 🌭


A classic favorite featuring a succulent sausage nestled in a soft bun, topped with mustard, ketchup, onions, and relish.

Tacos 🌮


Flavor-packed tacos with seasoned meat, fresh vegetables, and a zingy sauce, all wrapped in a warm tortilla.

WHICH CATEGORY OF SNACKS YOU LIKE ?


CHOOSE THE SNACK FROM THE ROULETTE




BURGER :




Burger XI



Burger XI




Burger XI




Burger XI


PIZZA :




Pizza hut



Pizza hut

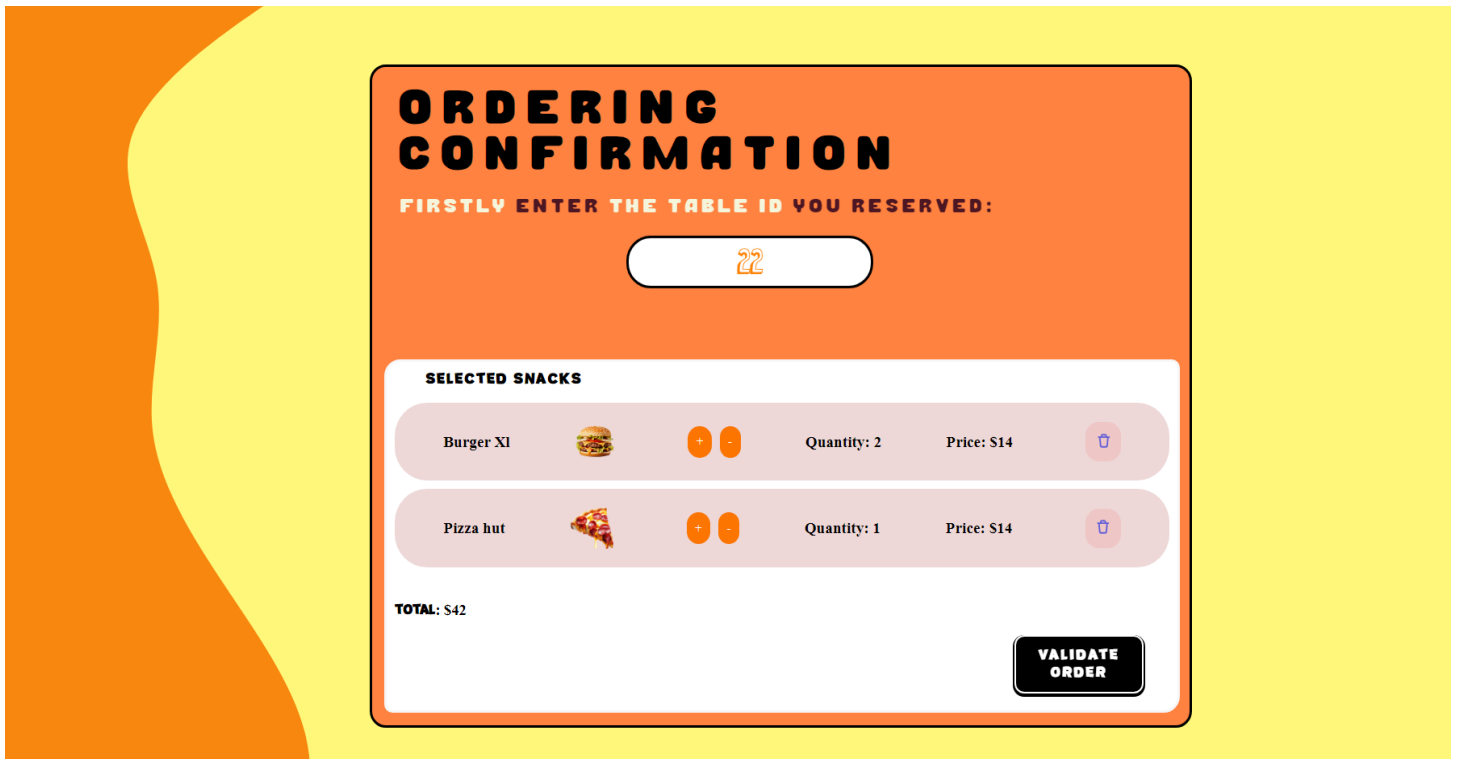


Pizza hut
 PRICE: \$14
 Choose



Pizza hut

8



TEST UNITAIRES: (JUNIT TEST)

TEST DES SERVICES DE COMMANDES :

Fonctions de test :

Ces tests couvrent les opérations essentielles pour la gestion des commandes dans votre application Spring Boot, garantissant le bon fonctionnement des méthodes de service associées à la manipulation des commandes.

Test GetAllCommandes :

Description : Vérifie que la méthode `getAllCommandes` retourne la liste de toutes les commandes existantes avec les détails corrects tels que l'ID, le total et l'état.

Test GetCommandeById :

Description : Vérifie que la méthode `getCommandeById` récupère une commande spécifique par son ID avec les détails corrects tels que le total et l'état.

Test UpdateCommande :

Description : Vérifie que la méthode updateCommande met à jour les détails d'une commande existante, tels que le total et l'état, et retourne la commande mise à jour.

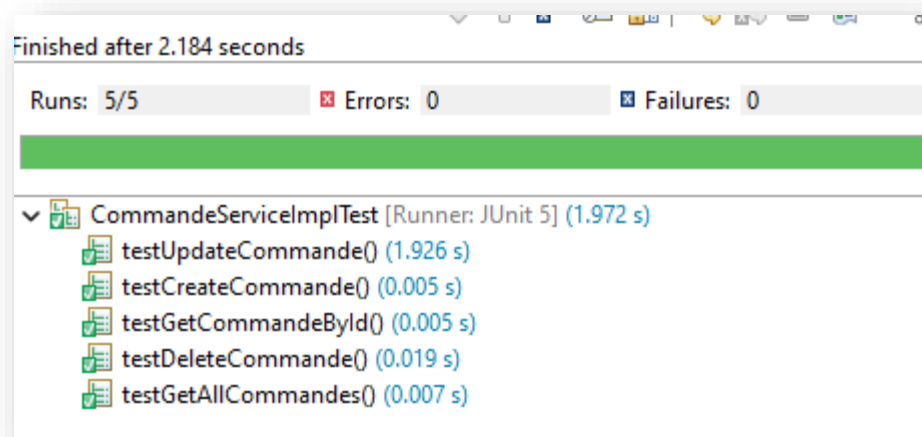
Test DeleteCommande :

Description : Vérifie que la méthode deleteCommande supprime une commande spécifique en appelant la méthode deleteById du référentiel.

Test CreateCommande :

Description : Vérifie que la méthode createCommande crée une nouvelle commande avec les détails appropriés tels que l'ID, le total et l'état, et retourne la commande créée.

JUnit tests :



Maven tests:

```
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 14.278 s
[INFO] Finished at: 2023-12-31T02:04:42+01:00
[INFO] -----
```

TEST DES SERVICES DE PRODUIT (SNACK) :

Fonctions de test :

Test GetAllCommandes :

Description : Vérifie que la méthode `getAllCommandes` retourne la liste de toutes les commandes existantes avec les détails corrects tels que l'ID, le total et l'état.

Test SaveSnack :

Description : Vérifie que la méthode `save` sauvegarde correctement un snack et retourne le snack sauvegardé avec les détails appropriés tels que l'ID, la catégorie et le prix.

Test GetAllSnacks :

Description : Vérifie que la méthode `getAllSnacks` récupère toutes les collations existantes depuis le référentiel avec les détails corrects tels que l'ID, la catégorie et le prix.

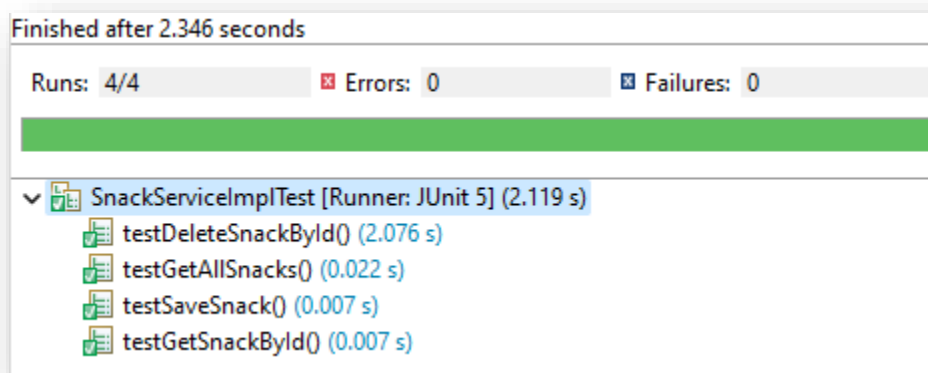
Test DeleteSnackById :

Description : Vérifie que la méthode `deleteSnackById` supprime une collation spécifique en appelant la méthode `deleteById` du référentiel avec le bon ID.

Test GetSnackById :

Description : Vérifie que la méthode `getSnackById` récupère correctement une collation spécifique par son ID depuis le référentiel avec les détails appropriés tels que l'ID, la catégorie et le prix.

JUnit tests :



Maven tests:

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.391 s -- in
Running eatitproject.eatitproject.services.CommandeServiceImplTest
Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.426 s -- in
Running eatitproject.eatitproject.services.SnackServiceImplTest
Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.102 s -- in

Results:

Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
```

SELENIUM

Scénario de Test :

L'objectif de ce test est de vérifier la fonctionnalité du processus de placement de commande sur notre plateforme. Le test implique une série d'actions imitant l'interaction d'un client avec le site pour sélectionner un plat, l'ajouter à la commande, choisir la quantité et placer la commande.

Résultat du Test :

Le test a été exécuté avec succès, et toutes les étapes spécifiées ont été réalisées sans rencontrer d'erreurs. L'alerte avec le message attendu a été confirmée avec succès.

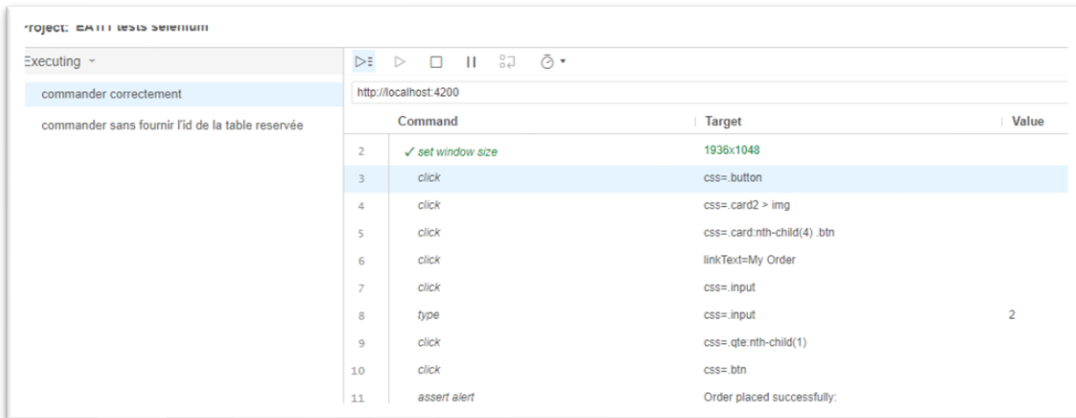
Observations :

Le site web a répondu conformément aux interactions de l'utilisateur.

Le processus de placement de commande, y compris la navigation dans la sélection des produits et l'interaction avec les boutons, a fonctionné comme prévu.

L'alerte confirmant le placement réussi de la commande a été affichée promptement.

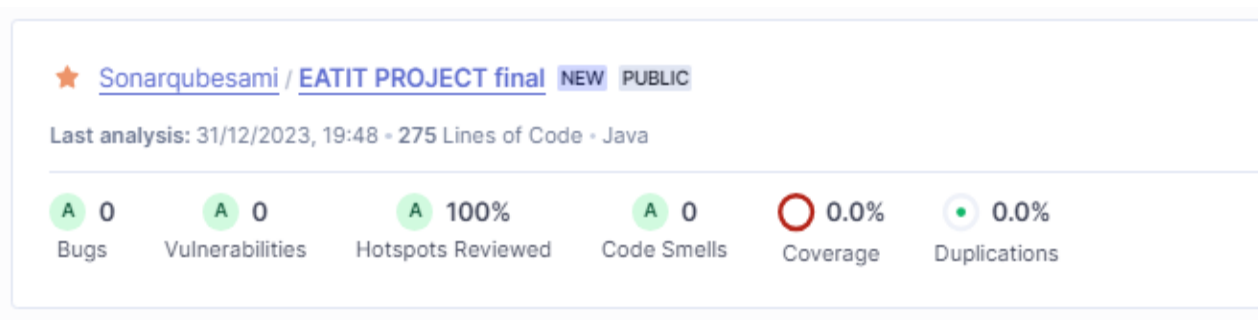
Scenario :



	DT	ORACLE	APPLICATION	OBSERVATION
PlaceOrder(Order) : sans fournir l'id de la table	Erreur	Erreur : (non enregistrement de la commande)	Alerte(Please enter the table number before submitting the order.)	Valide
PlaceOrder : (Order)	Order object	Order Object sent to BackEnd) +alert de sauvegarde de commande	Alerte(Order placed successfully)+http request /post(order)	Valide

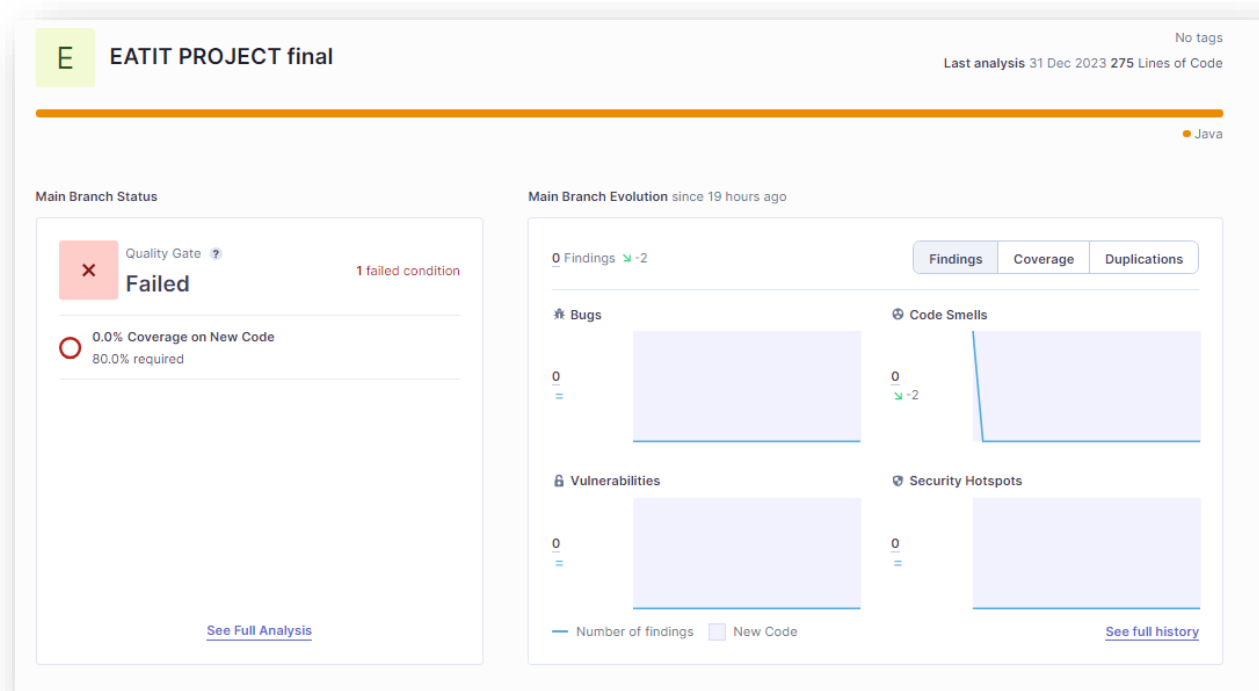
SonarQube pour améliorer la qualité globale de votre code Spring Boot(Back-end):

- 1- Configuration du Projet ;
- 2- Intégration et Configuration ;
- 3- Exécution de Sonar Scanner ;
- 4- Rapports de Qualité du Code ;



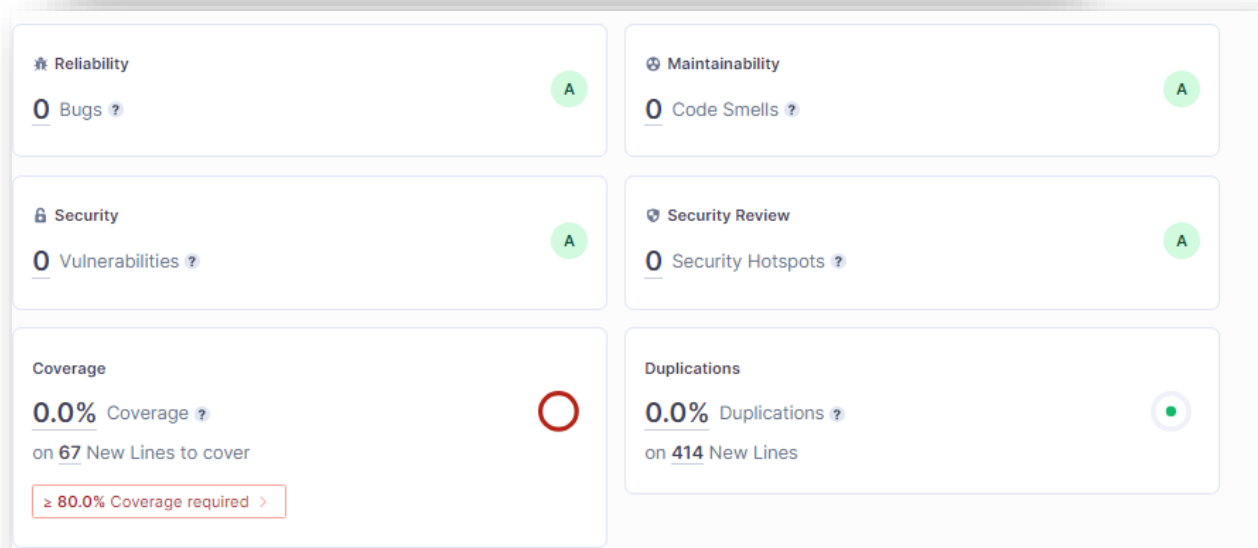
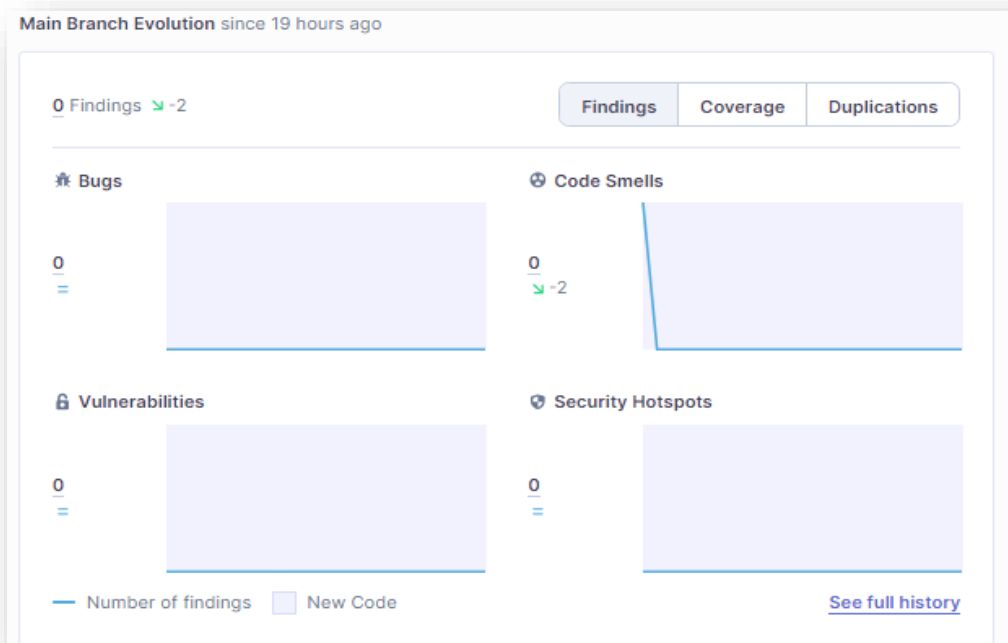
Sonar Cloud Dashboard :

Observation : Même si j'ai exécuté les tests unitaires, la couverture de code par les tests unitaires reste 0%



Correction des Problèmes Signalés par Sonar Qube :

Optimisation et Correction du source code après les scans répétitifs.



Maintainability :



Security :



Vulnerabilité :

Vulnerabilities	0
Rating	A

Dockerfile :

```

1 # Use an official OpenJDK runtime as a parent image
2 FROM openjdk:11-jre-slim
3
4 # Set the working directory to /app
5 WORKDIR /app
6
7 # Copy the JAR file into the container at /app
8 COPY target/eatitproject.jar eatitproject.jar
9
10 # Expose the port that your application runs on
11 EXPOSE 8088
12
13 # Define the command to run your application
14 CMD ["java", "-jar", "eatitproject.jar"]
15

```

Docker image :

```

C:\Users\vansa\OneDrive\Documents\eatitprojectdocker>docker build -t eatitproject.jar .
[+] Building 44.7s (8/8) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 422B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/openjdk:11-jre-slim
=> [1/3] FROM docker.io/library/openjdk:11-jre-slim@sha256:93af7df2308c5141a751c4830e6b6c5717db102b3b31f012ea29d842dc4f2b02
=> => resolve docker.io/library/openjdk:11-jre-slim@sha256:93af7df2308c5141a751c4830e6b6c5717db102b3b31f012ea29d842dc4f2b02
=> => sha256:12cca292b13cb58fadde25af113ddc4ac3b0c5e39ab3f1290a6ba62ec8237afd 212B / 212B
=> => sha256:93af7df2308c5141a751c4830e6b6c5717db102b3b31f012ea29d842dc4f2b02 549B / 549B
=> => sha256:884c08d0f406a81aeb5786932abaf399c335b997da7eea6a30cc51529220b66 1.16kB / 1.16kB
=> => sha256:764a04af3eff09cc6a29bcc19cf6315dbea455d7392c1a588a5deb331a929c29 7.55kB / 7.55kB
=> => sha256:1efc276f4ff952c055dea726cfc96ec6a4fdb8b62d9eed816bd2b788f2860ad7 31.37MB / 31.37MB
=> => sha256:a2f2f93da48276873890ac821b3c991d53a7e864791aaf82c39b7863c908b93b 1.58MB / 1.58MB
=> => sha256:d73cf48caaac2e45ad76a2a9eb3b311d0e4eb1d804e3d2b9cf075a1fa31e6f92 46.04MB / 46.04MB
=> => extracting sha256:1efc276f4ff952c055dea726cfc96ec6a4fdb8b62d9eed816bd2b788f2860ad7
=> => extracting sha256:a2f2f93da48276873890ac821b3c991d53a7e864791aaf82c39b7863c908b93b
=> => extracting sha256:12cca292b13cb58fadde25af113ddc4ac3b0c5e39ab3f1290a6ba62ec8237afd
=> => extracting sha256:d73cf48caaac2e45ad76a2a9eb3b311d0e4eb1d804e3d2b9cf075a1fa31e6f92
=> [internal] load build context
=> => transferring context: 71.35MB
=> [2/3] WORKDIR /app
=> [3/3] COPY target/eatitproject.jar eatitproject.jar
=> => exporting to image
=> => exporting layers
=> => writing image sha256:45d608a7a6d812ae4c7b0ea94454ce04af6ff3f16b540375d26f7e57d8b100f0
=> => naming to docker.io/library/eatitproject.jar

```

View build details:

C:\Users\vansa\OneDrive\Documents\eatitprojectdocker>

```

C:\Users\vansa\OneDrive\Documents\eatitprojectdocker>docker build -t eatitproject-3.2.1.jar .
[+] Building 0.1s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 427B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/rsunix/yourkit-openjdk17:latest
=> CACHED [1/3] FROM docker.io/rsunix/yourkit-openjdk17
=> [internal] load build context
=> => transferring context: 2B
=> [2/3] WORKDIR /app
=> ERROR [3/3] COPY target/eatitproject.jar eatitproject.jar
-----
> [3/3] COPY target/eatitproject.jar eatitproject.jar:
Dockerfile:8
-----
6 |
7 | # Copy the JAR file into the container at /app
8 | >>> COPY target/eatitproject.jar eatitproject.jar
9 |
10 | # Expose the port that your application runs on
-----
ERROR: failed to solve: failed to compute cache key: failed to calculate checksum of ref 55f560c7-82cd-41b4-b48e-34286e9a89af::pjdp817w8
d7yl9huahlsvyqj: "/target/eatitproject.jar": not found

View build details:
C:\Users\vansa\OneDrive\Documents\eatitprojectdocker>

```





```








C:\Users\vansa\OneDrive\Documents\eatitprojectdocker>docker build -t eatit-image .
2024/01/20 00:55:21 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 3.1s (8/8) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 406B
=> [internal] load metadata for docker.io/rsunix/yourkit-openjdk17:latest
=> [1/3] FROM docker.io/rsunix/yourkit-openjdk17
=> [internal] load build context
=> => transferring context: 71.35MB
=> CACHED [2/3] WORKDIR /app
=> [3/3] COPY target/eatit.jar eatit.jar
=> exporting to image
=> => exporting layers
=> => writing image sha256:3d5e71da1986b76596cb332b35bc1ef004087e8246447cd6b896cf9604d20aaa
=> => naming to docker.io/library/eatit-image
View build details: █

```

Docker DesktopUpgrade plan

Search for images, containers, volumes, extensions and more...Ctrl+K























Images

[Give feedback](#)


LocalHubArtifactoryEARLY ACCESS


1.41 GB / 1.64 GB in use8 ImagesLast refresh: 2 hours ago

Search

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input checked="" type="checkbox"/>	eatit-image 3d5e71da1986	latest	In use	48 minutes ago	636.59 MB	  
<input type="checkbox"/>	eatitproject.jar 45d608a7a6d8	latest	In use	1 hour ago	294.6 MB	  
<input type="checkbox"/>	rabbitmq 51d7e67ae1b6	latest	Unused	14 days ago	216.74 MB	  
<input type="checkbox"/>	jenkins/jenkins 41e27c2a574b	Its	In use	1 month ago	485.58 MB	  
<input type="checkbox"/>	rabbitmq 138bc578945f	3.12.9-management	Unused	8 months ago	246.3 MB	  
<input type="checkbox"/>	rsunix/yourkit-openjdk17 f5d050c7ce21	latest	Unused	1 year ago	565.25 MB	  

Showing 8 items

RAM 1.96 GBCPU 1.04%Not signed in

v4.26.1