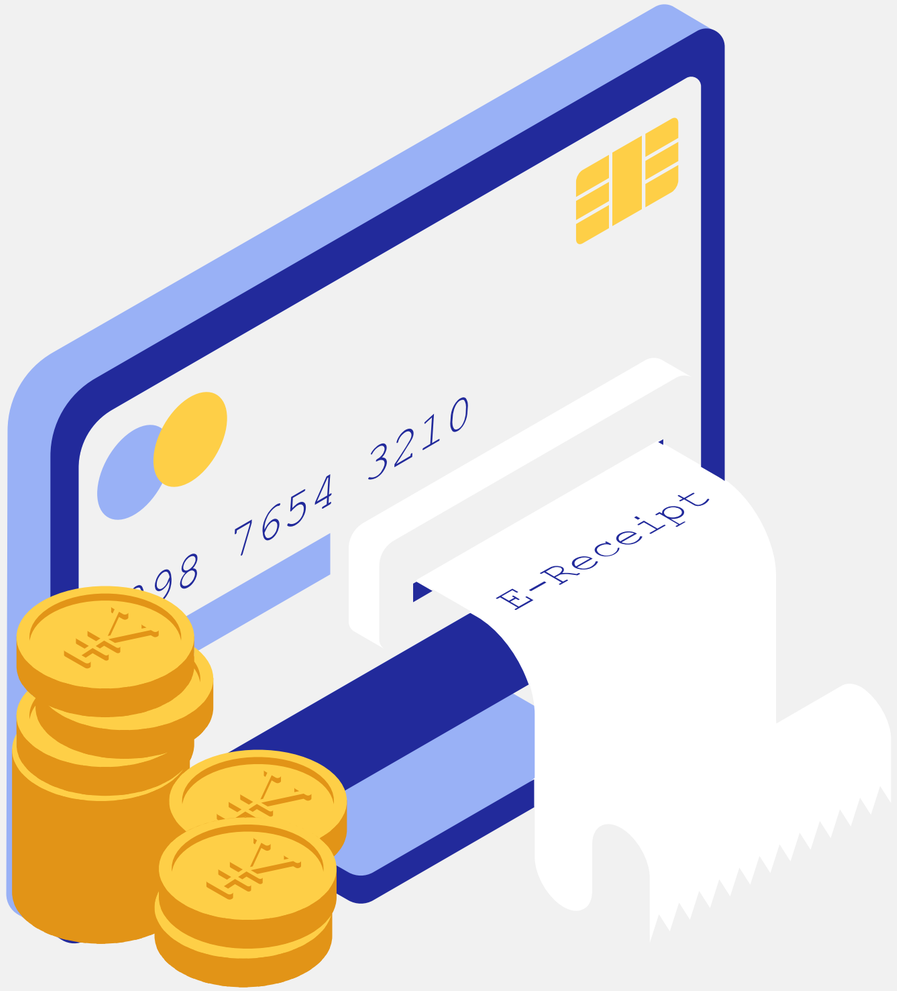


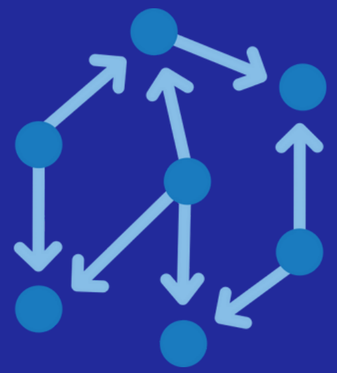
Lorenzo Marcolli - mat. 08302A

e-Commerce Saga Orchestration Implementation

Progetto di Cloud Computing Technologies

Saga orchestration

L'obiettivo del progetto è la gestione di un'ordine in un eCommerce utilizzando il pattern Saga Orchestration



Pattern di progettazione che ha l'obiettivo di gestire transazioni orchestrando diversi microservizi.

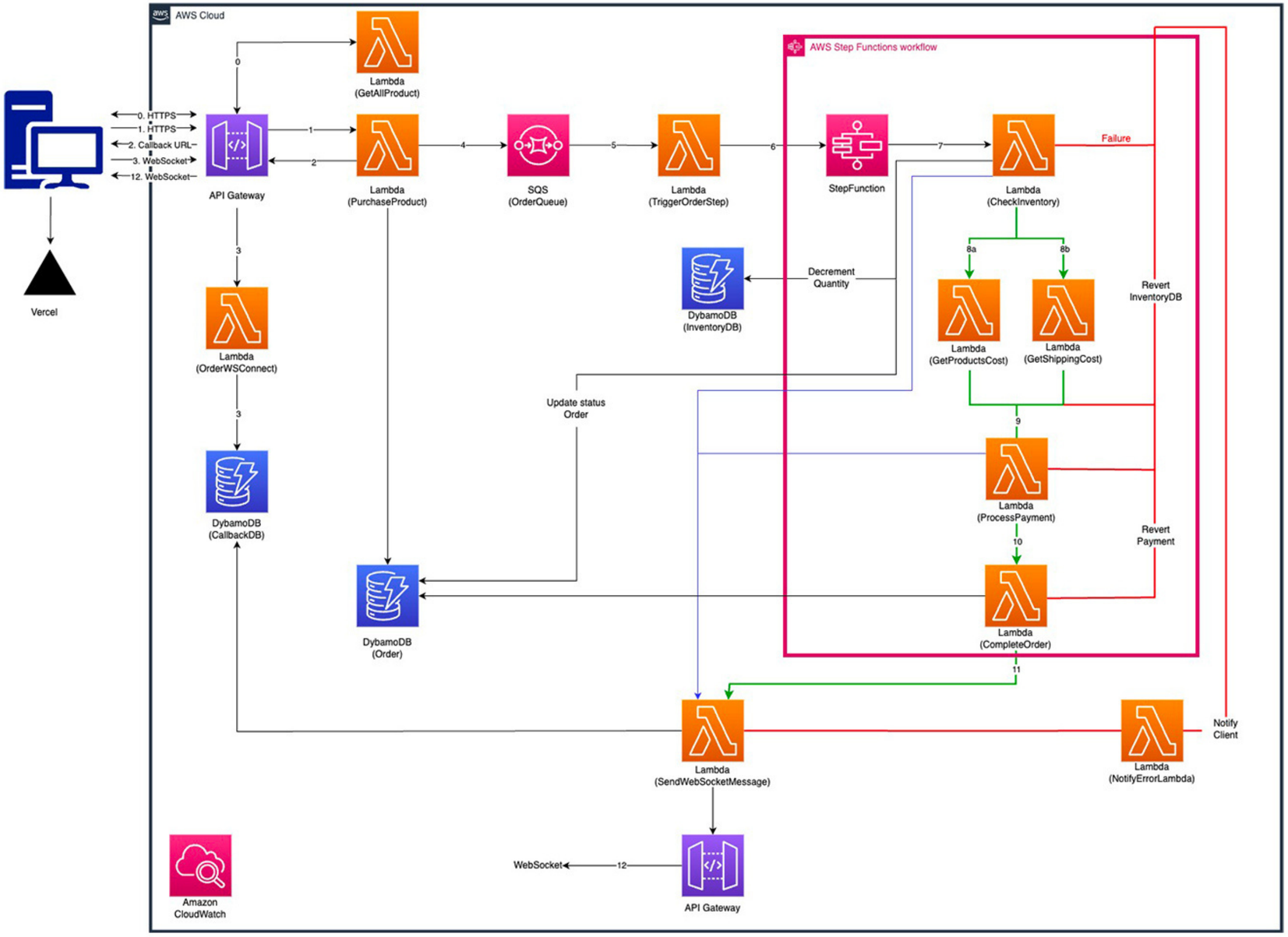
Le transazioni vengono suddivise in più fasi, ognuna delle quali rappresenta una singola operazione all’interno di un servizio.

Se una delle operazioni fallisce, l’orchestratore può eseguire azioni di compensazione per annullare le operazioni precedenti.

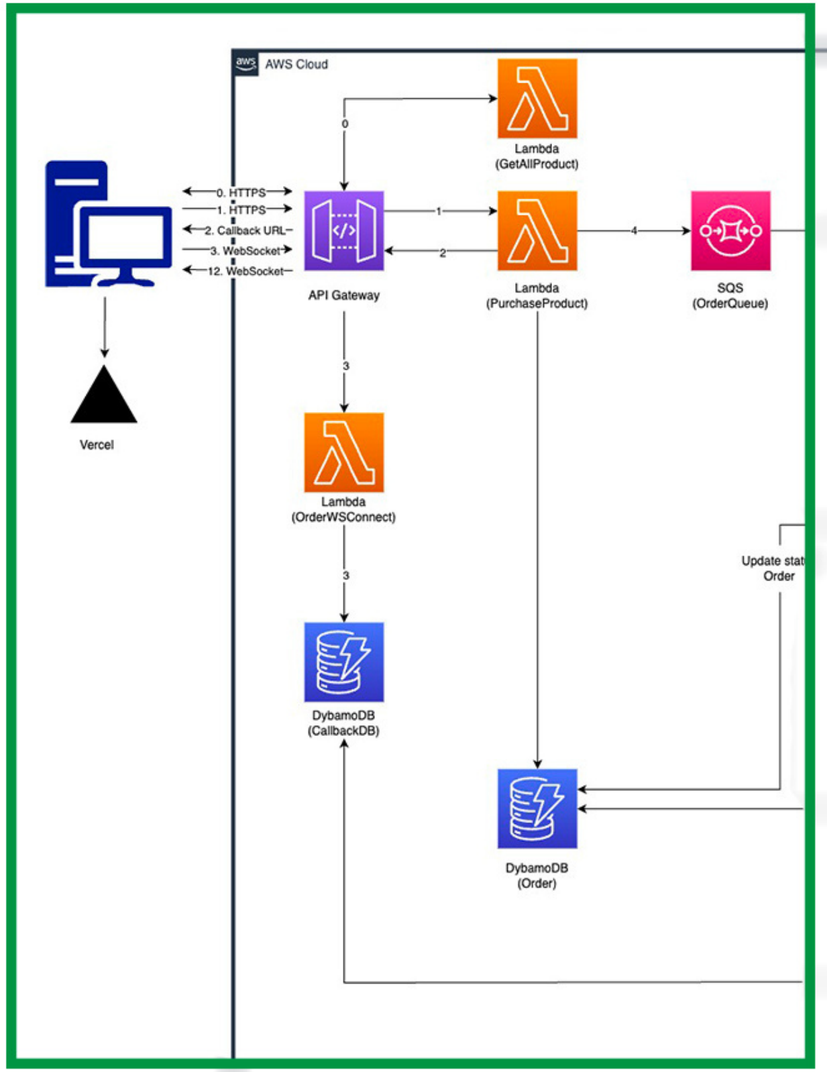
E' stato utilizzato AWS come piattaforma cloud.

aws

Workflow

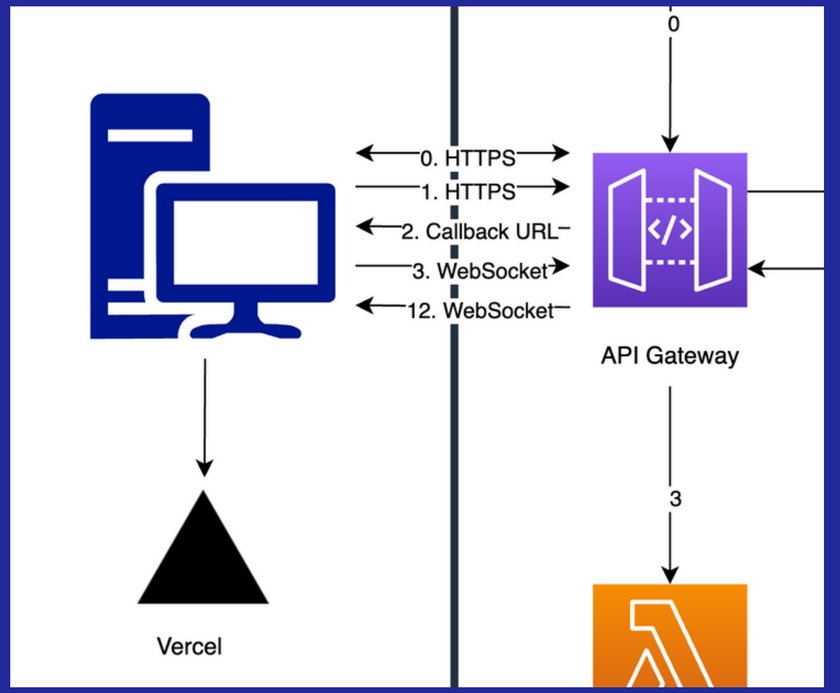

AWS Cloud
Lambda (GetAllProduct)
AWS Step Functions workflow
0. HTTPS
1. HTTPS
2. Callback URL
3. WebSocket
12. WebSocket
Vercel
API Gateway
Lambda (PurchaseProduct)
SQS (OrderQueue)
Lambda (TriggerOrderStep)
StepFunction
Lambda (CheckInventory)
Failure
DybamoDB (InventoryDB)
Decrement Quantity
Lambda (GetProductsCost)
Lambda (GetShippingCost)
Revert InventoryDB
Lambda (OrderWSConnect)
Update status Order
DybamoDB (CallbackDB)
Lambda (ProcessPayment)
Revert Payment
DybamoDB (Order)
Lambda (CompleteOrder)
Lambda (SendWebSocketMessage)
Lambda (NotifyErrorLambda)
Notify Client
Amazon CloudWatch
WebSocket
API Gateway

Workflow


AWS Cloud
0
Lambda
(GetAllProduct)
0. HTTPS
1. HTTPS
2. Callback URL
3. WebSocket
12. WebSocket
1
2
4
API Gateway
Lambda
(PurchaseProduct)
SQS
(OrderQueue)
Vercel
3
Lambda
(OrderWSConnect)
3
DybamoDB
(CallbackDB)
DybamoDB
(Order)

Client

La WebApp

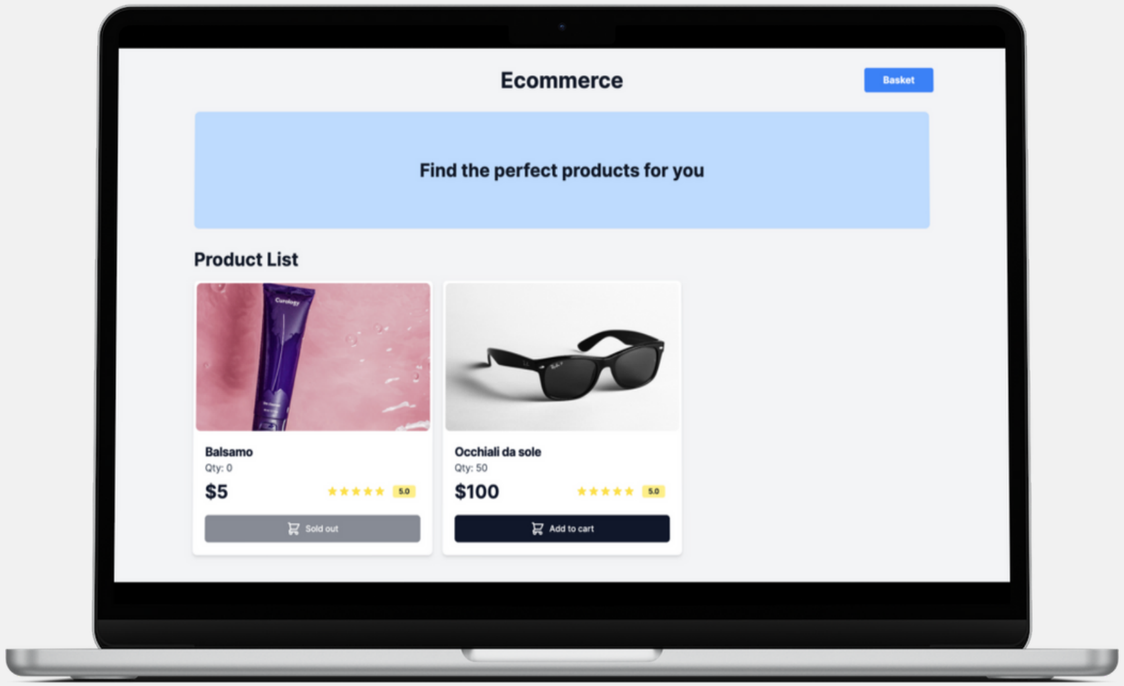


Webapp

Il Client è quell’applicazione con la quale un utente può effettuare un ordine, andando così ad iniziare l’intero flow di acquisto.

E' stata realizzata una semplice piattaforma dal quale un utente può effettuare un ordine utilizzando:

- Next.js
- Tailwindcss
- Typescript



Link

Vercel

Piattaforma di deployment e hosting per applicazioni web che si integra perfettamente con Next.js

E' stato scelto di utilizzare Vercel come piattaforma di hosting per la WebApp per:

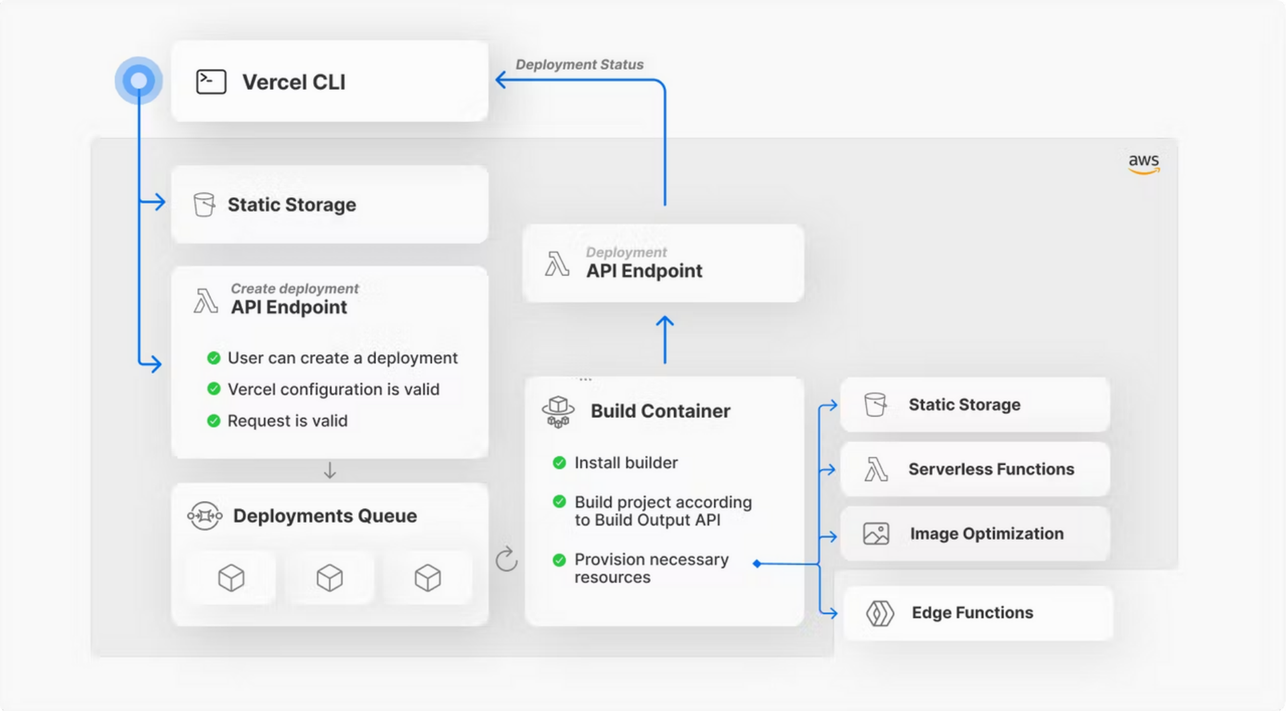
- Performance
- Intergrazione CI/CD
- Scalabilità

Architettura di Vercel

Fase di Deploy

La fase di deploy su Vercel è suddivisa in diversi step:

- Creazione di un deployment da parte di un utente partendo dal codice
- Tutta la fase di build viene effettuata in un Build Container
- Il nuovo deploy è disponibile tramite i CDN di Vercel

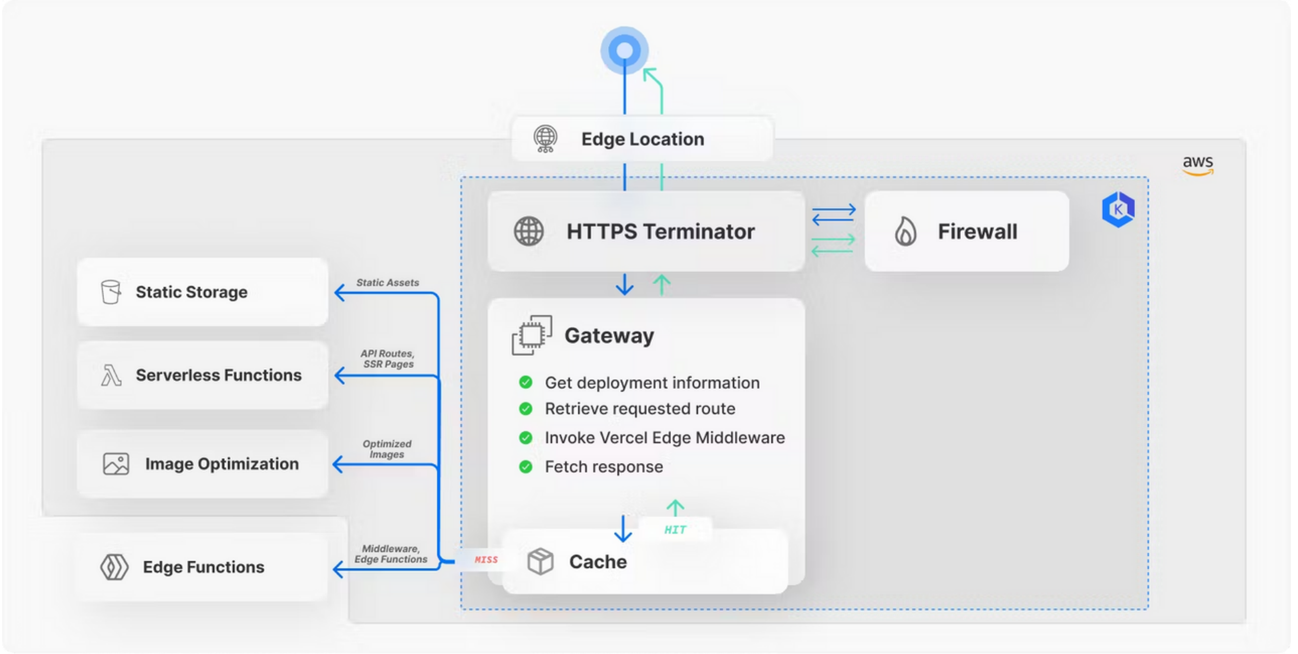


Architettura di Vercel

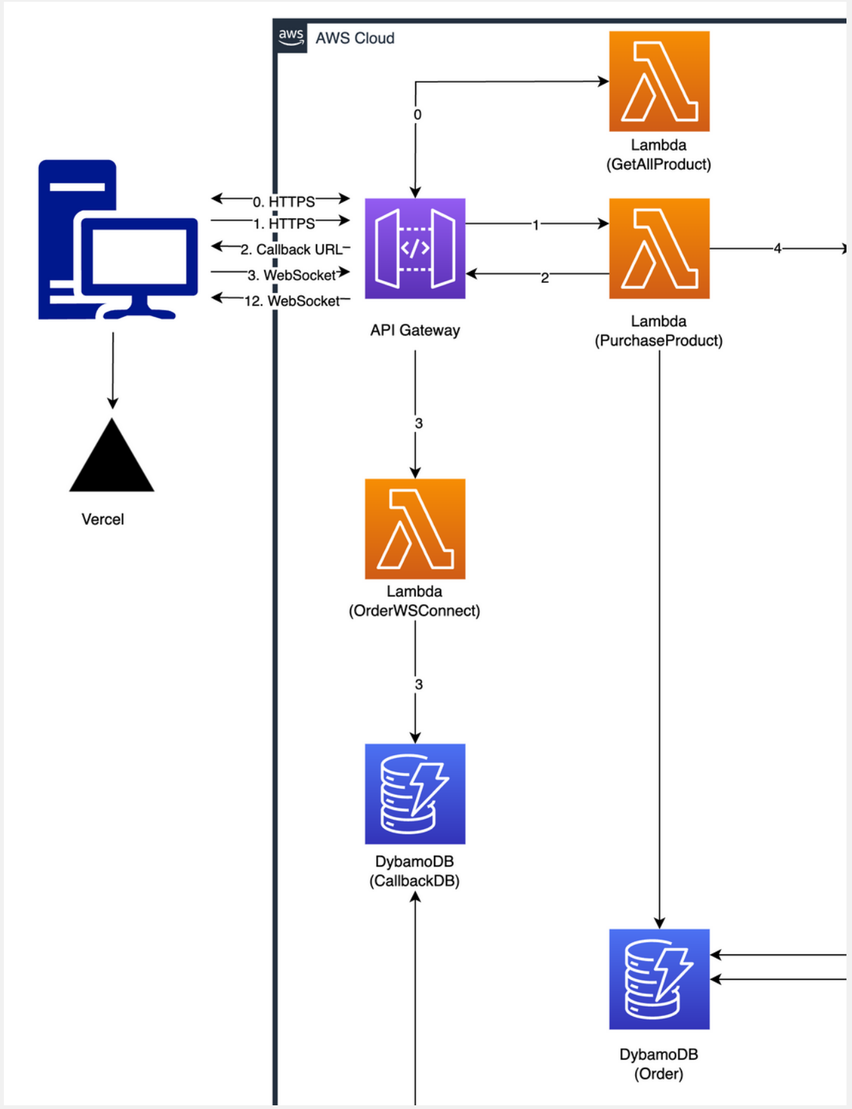
Fase di Request

La fase di request su Vercel è suddivisa in diversi step:

- Viene effettuato un lookup DNS per trovare l'indirizzo IP
  - Vercel utilizza indirizzi IP anycast
- La richiesta entra in un cluster Kubernetes
  - Viene ispezionata tramite Firewall
  - Viene recuperato il deployment richiesto
  - Vengono eseguite funzioni middleware se presenti
  - Viene recuperata la risorsa e ritornata al client



Flow Client



0. Vengono recuperati tutti i prodotti presenti nel database GET

Il client tramite chiamata REST API recupera tutti i prodotti presenti nel database, tramite la lambda GetAllProduct

1. Viene effettuato un ordine POST

Il client tramite REST API invia un payload contenente i dati dell'ordine che vuole effettuare.

2. Risposta contenente ID dell'ordine e URL per la connessione alla websocket

Viene creato l'ordine, salvandolo su una tabelle di DynamoDB con status "pending". Viene ritornato al client l'ID del nuovo ordine e un URL per la connessione ad una WebSocket.

3. Connessione alla WebSocket CONNECT

Il client si connette all'URL di WebSocket per poter ricevere aggiornamenti sullo stato dell'ordine. La lambda OrderWSConnect salverà il match tra ID dell'ordine e ID della connessione.

Servizi AWS

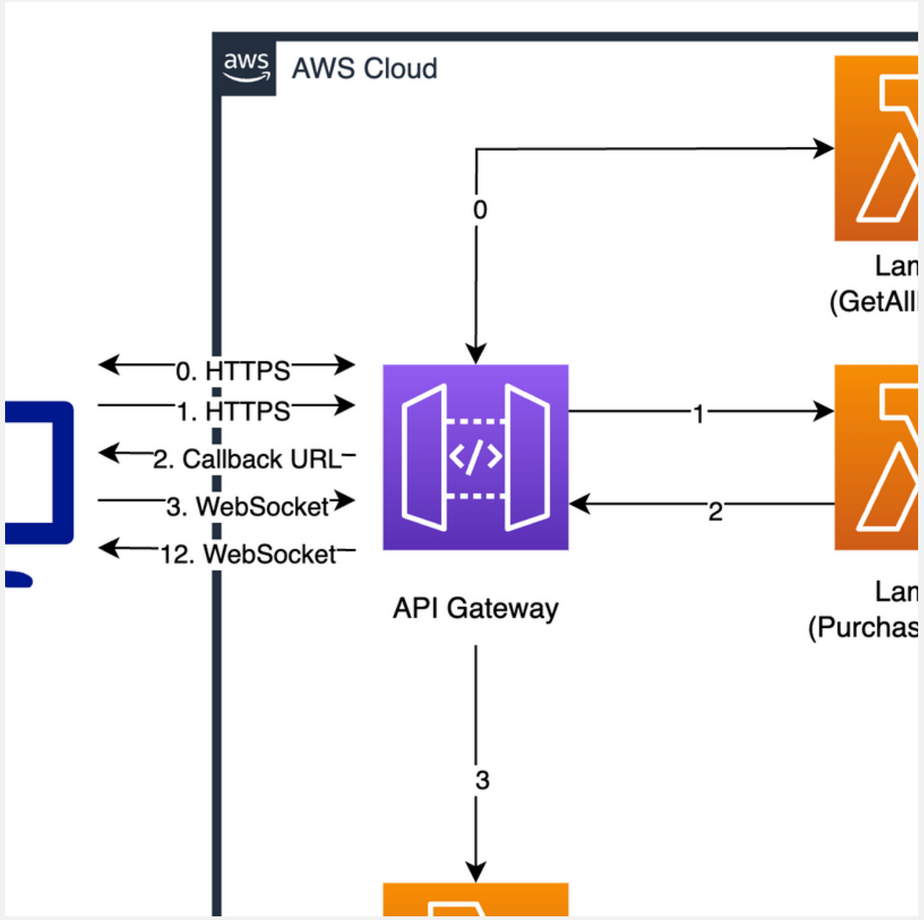
Api Gateway

API Gateway è un servizio di gestione delle API. Consente di creare, pubblicare, gestire e proteggere le API RESTful e WebSocket.

Nel nostro caso viene utilizzata per due soluzioni:

1. Trigger per lambda
2. Websocket

L'API Gateway può gestire fino a 10.000 richieste al secondo. Se abbinata con Lambda anch'essa scalerà in automatico per far fronte al numero di richieste in entrata.



Le API saranno disponibili in diverse località fisiche in tutto il mondo. La distribuzione multi-regione offre diversi vantaggi, tra cui la ridondanza e la tolleranza ai guasti.

Lambda

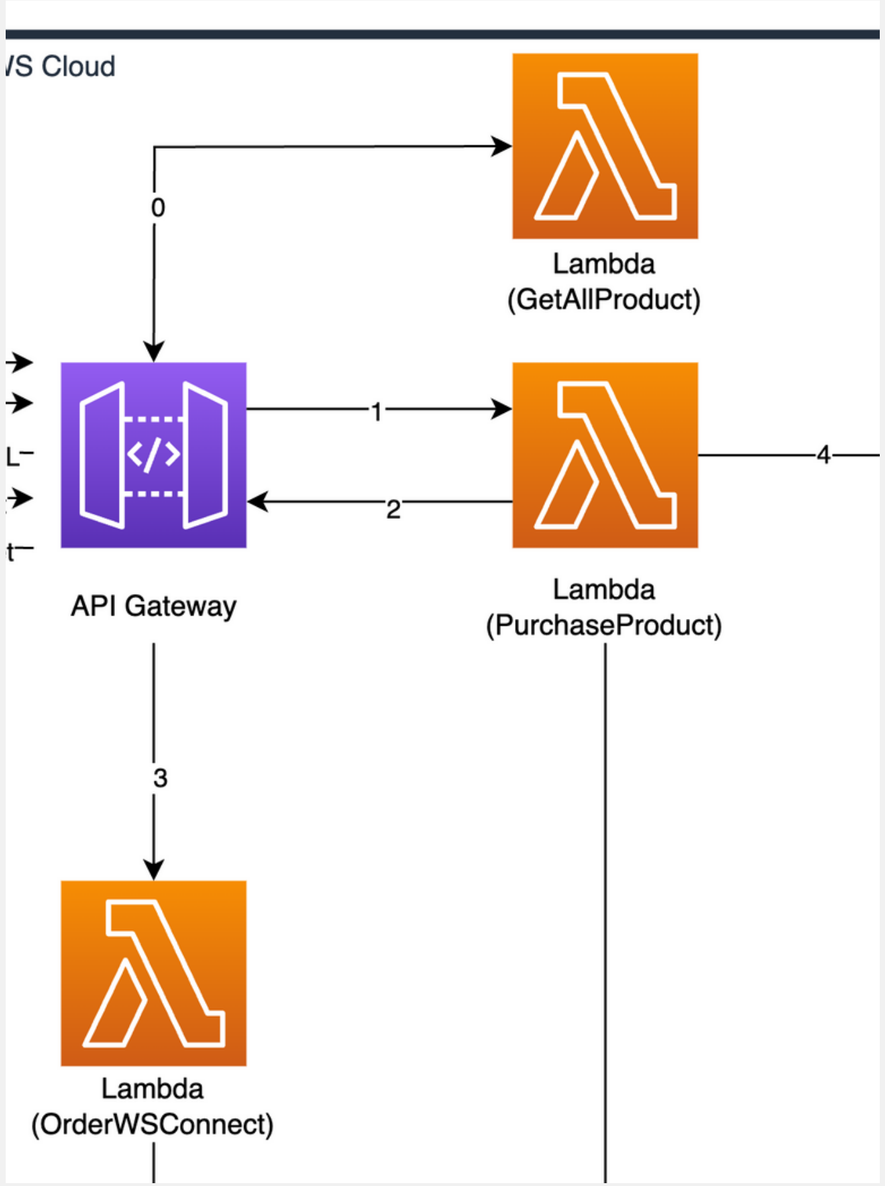
AWS Lambda è un servizio di calcolo serveless.

Supporta diversi linguaggi di programmazione, nel nostro caso è stato utilizzato:

- Javascript con Node.js
- Python

Alcune delle caratteristiche interessanti sono:

- Versioning
- Scalabilità automatica
- Alta disponibilità



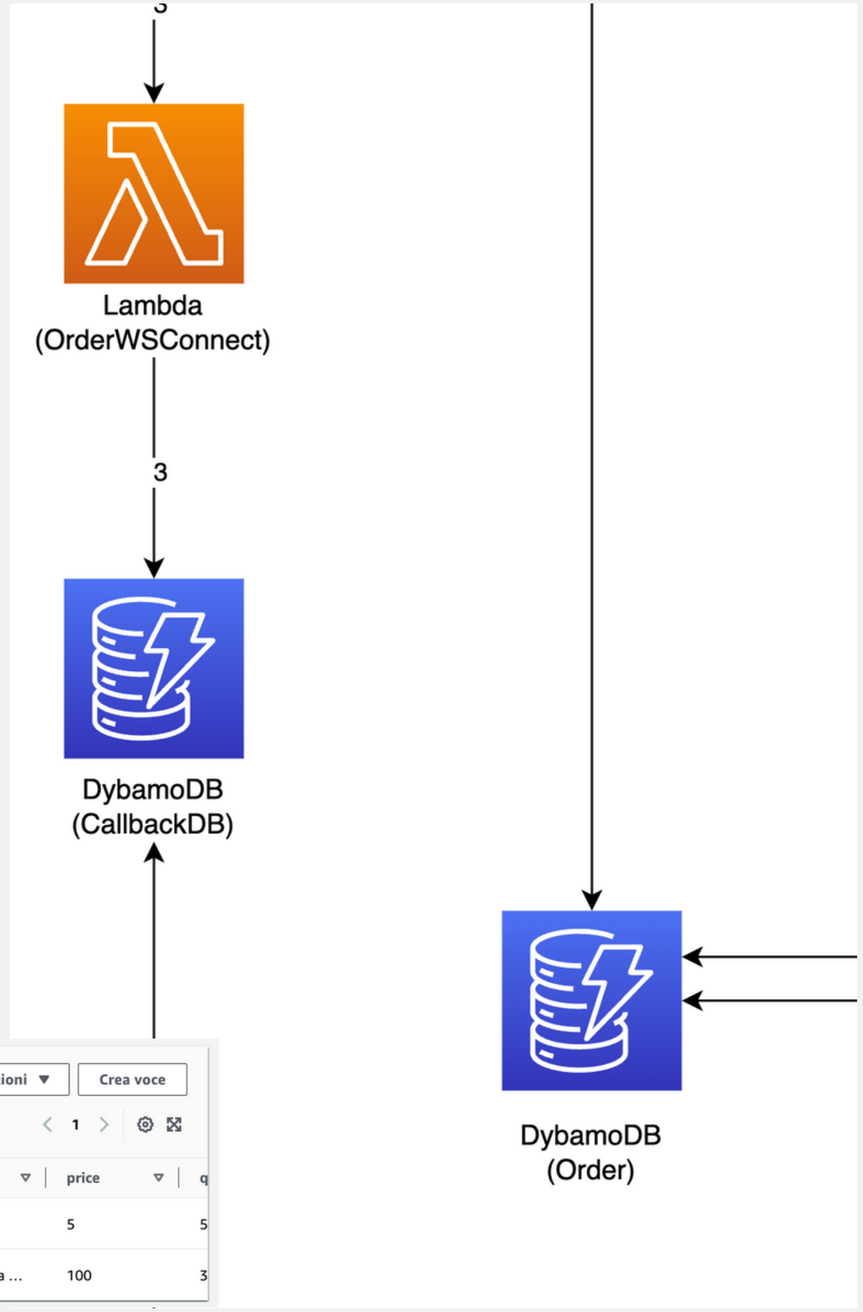
DynamoDB

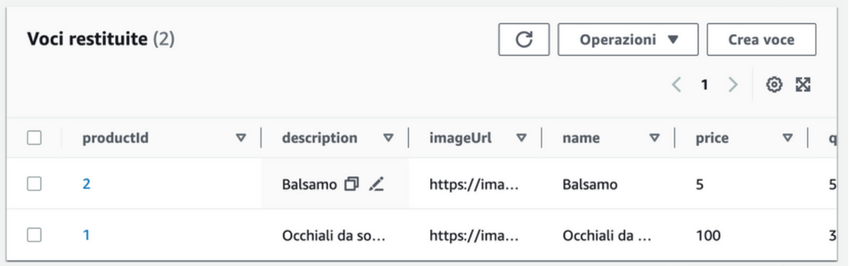
DynamoDB è un servizio di database gestito e scalabile progettato per archiviare e recuperare grandi quantità di dati in modo affidabile e ad alte prestazioni

- Database chiave-valore
- Scalabilità automatica
  - Partizionamento dei dati
  - Provisioned Throughput
- Ridondanza dei dati

Nel progetto è stato utilizzato per salvare:

- Ordini
- Inventario con la disponibilità
- Abbinamento tra connection ID e ID ordine




Voci restituite (2)

Operazioni | Crea voce

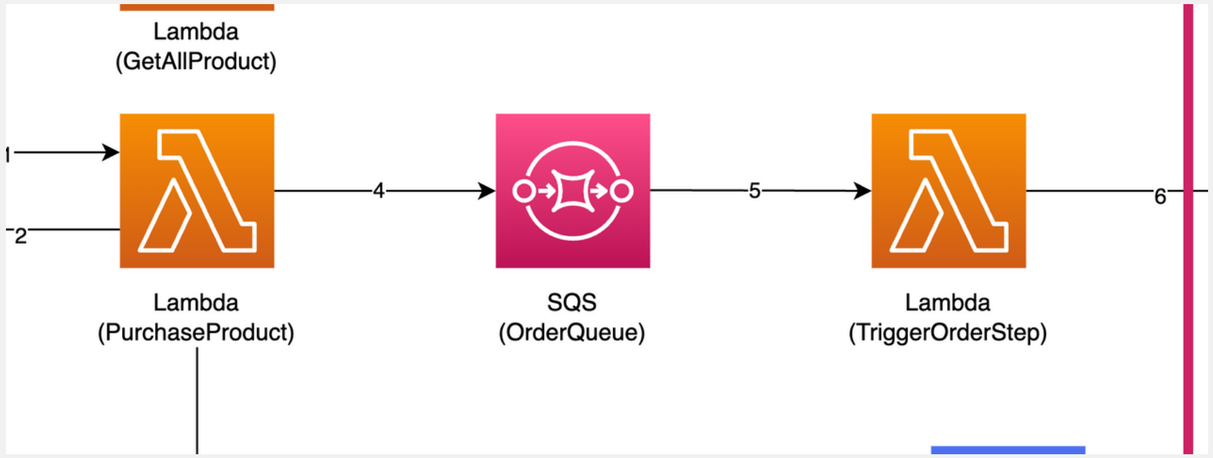
1

productId	description	imageUrl	name	price	q
2	Balsamo	https://ima...	Balsamo	5	5
1	Occhiali da so...	https://ima...	Occhiali da ...	100	3

Simple Queue Service (1/2)

SQS consente di inviare, memorizzare e ricevere qualsiasi volume di messaggi tra componenti software senza perdita di dati e indipendentemente dalla disponibilità di altri servizi.

- E un servizio di accodamento messaggi completamente gestito
- Disaccoppiato da altri componenti che possono quindi fallire indipendentemente
- Garantisce una comunicazione affidabile tra componenti software distribuiti e microservizi



Simple Queue Service (2/2)

Amazon SQS offre due tipi di coda:

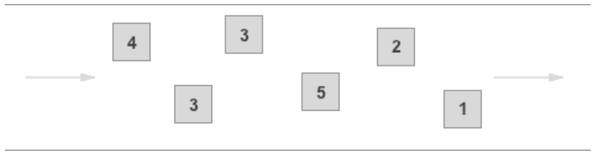
1. Code standard
2. Code FIFO

- Ordine garantito
- Affidabilità
- Fault tollerance

Scelta

Standard

- Consegna del mesaggio: "miglior ordine possibile"
  - Ordine di invio diverso dalla ricezione
- Il messaggio viene consegnato "At-Least-Once", ma potrebbe essere venga consegnato più volte



FIFO

- Consegna del mesaggio: "exactly-once"
  - Ordine di invio uguale all'ordine di ricezione
- Il messaggio rimane disponibile fino a quando un client non lo elabora e lo cancella
- Non vengono introdotti duplicati nella coda.

IAM

Identity and Access Management, è un framework o una serie di procedure e tecnologie utilizzate per la gestione delle identità e degli accessi su AWS.

Con le policy è possibile definire con granularità quali servizi e risorse possono interagire tra loro.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "LambdaPermissions",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "arn:aws:lambda:region:account-id:function:function-name"
      ]
    },
    {
      "Sid": "S3Permissions",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/*"
      ]
    }
  ]
}
```

Conclusioni

Utilizzando Amazon AWS e il pattern Saga orchestration:

Scalabilità

L'utilizzo della saga orchestration su cloud consente di creare un'architettura di microservizi altamente scalabile in base ai picchi di traffico.

Coordinazione dei flussi di lavoro

Semplifica la coordinazione dei flussi di lavoro complessi che coinvolgono più microservizi.

Affidabilità e tolleranza ai guasti

I servizi e le risorse possono essere distribuiti su più zone o regioni geografiche, riducendo il rischio di interruzioni del servizio.

Integrazione con altri servizi

Il cloud offre una vasta gamma di servizi aggiuntivi che possono integrarsi facilmente con l'architettura di saga orchestration

Grazie

lorenzo.marcolli@studenti.unimi.it

Repository del progetto