Coca-Cola Vending Pass

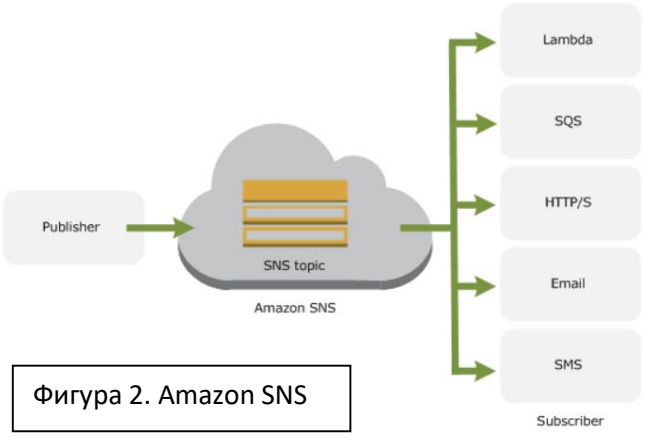
… or how things go better with *Step Functions*

Coca-Cola Vending Pass е дигитална карта за лоялност, която всеки може да добави към своето мобилно портмоне и да се включи в играта чрез покупки от избрани Кока-Кола вендинг машини. Самата игра протича по следния начин (*Фигура 1*): Всеки път, в който клиент заплати своята кока-кола чрез дигиталната си карта, направеното заплащане се отбелязва. След като направи 10 такива заплащания, той печели безплатна напитка.

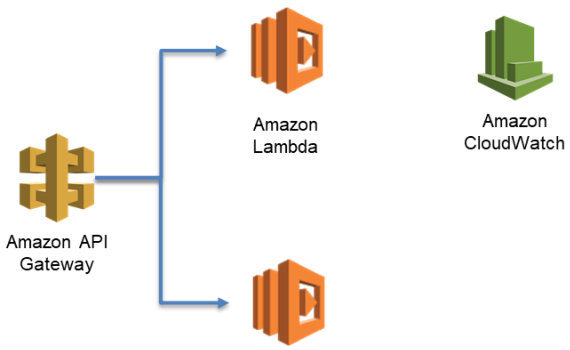
Фигура 1. Работа на Coca-Cola Vending Pass

Каква обаче е архитектурата на едно такова приложение?

Накратко в началото то е било разработено като комбинация от **SNS** (**(Amazon) Simple Notification Service**), което представлява уеб услуга, разработена от Амазон, която координира и навигира изпращането или получаването на съобщения до абонирани за нея клиенти (клиентите са два вида- *publishers* и *subscribers*, още познати като *producers* и *consumers*), и **AWS Lambda Functions** (**AWS=Amazon Web Services**), функции, които позволяват изпълнението на код без осигуряването на сървъри. (*Фигура 2 И Фигура 3*).



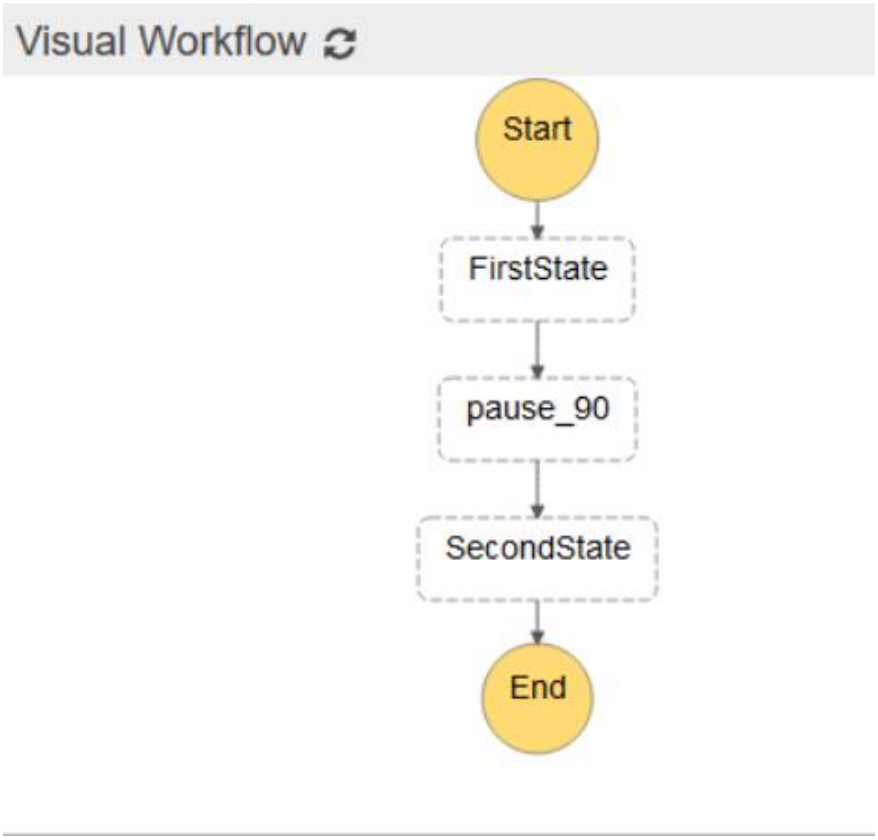
Фигура 2. Amazon SNS



Фигура 3. Amazon Lambda Functions

Тази комбинация обаче „се нуждаела“ от малко помощ, която „получила“ от съществуващ вече *backend* код, благодарение на който се преброяват точките от направените покупки и се подновява резултата на клиента. За съжаление обаче кодът реагирал по-бавно от очакваното и с това отнемал време, докато се изпълни, което, от своя страна, довело до пропускането на изясняването на някои детайли, което впоследствие довело до объркването на участниците във виртуалната игра. Най-интуитивното решение, естествено, било просто: модифициране на Ламбда кода чрез добавяне на *90-секундно изчакване* между двете фази. Това решило поставения проблем, но „изяло“ процесорно време (по принцип приложението на Ламбда функциите се базира на продължителността на заявките, обикновено те се подават в 100 ms-ни интервали). (Повече за Ламбда функциите може да научите от тук: https://www.youtube.com/watch?time_continue=7&v=eOBq__h4OJ4&feature=emb_title).

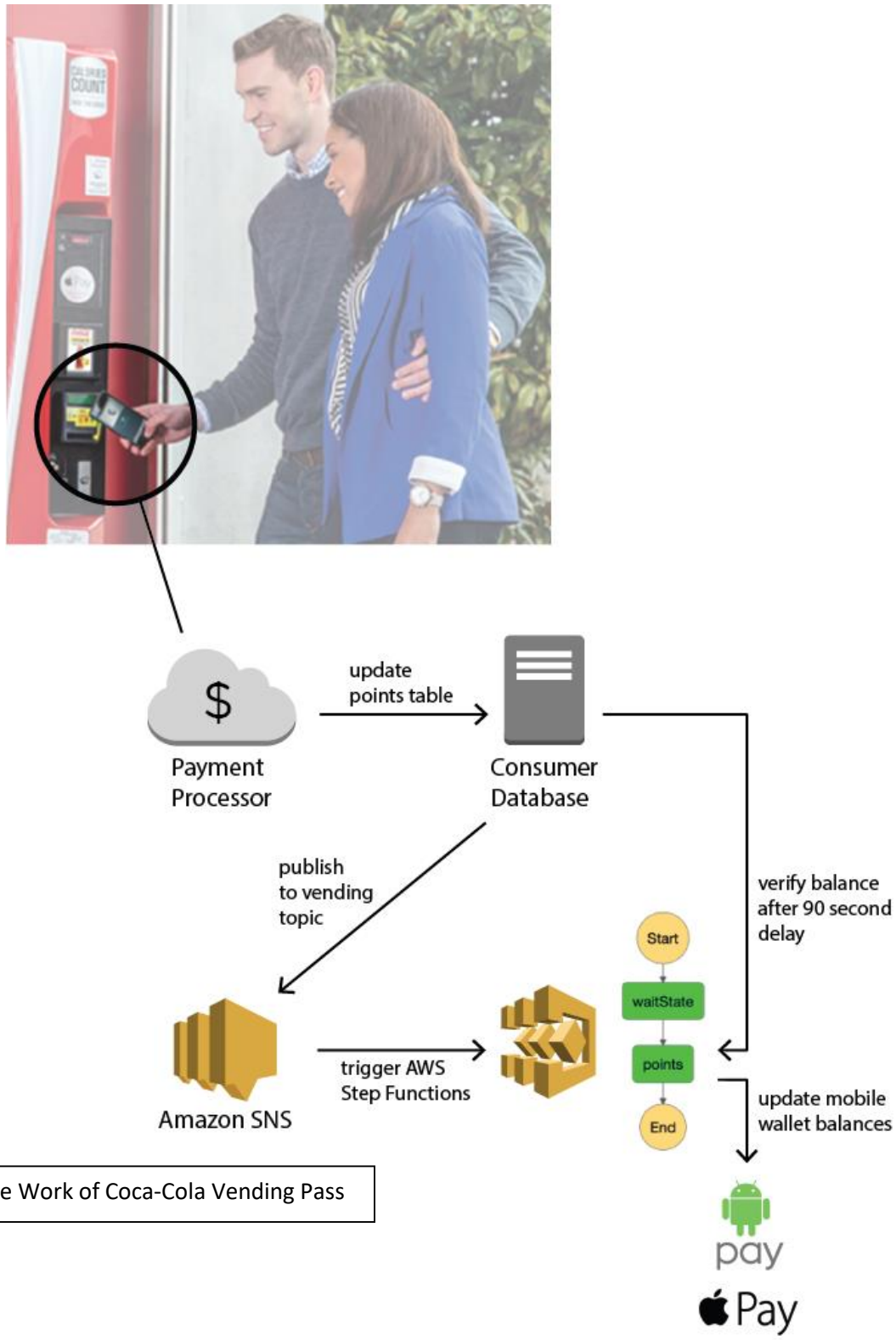
Затова с цел да направи своето решение по-рентабилно екипът на Кока-Кола решил да използва т.нар. **AWS Step Functions** (функции, които координират компонентите на разпределени системи и микросървиси в мащаб, използвайки визуални работни процеси, които лесно се изграждат). Екипът изградил доста проста машина на състоянията, за да улеснят своята бизнес логика и намалят разходите, която изглежда така:



Фигура 4. Машина на състоянията

Първото състояние (**FirstState**) и второто състояние (**SecondState**) – *Фигура 4*, са познати като състояния на задачите (**Task States**). Те извикват правилните Ламбда функции, докато Step функциите (**Step Functions**) имплементират *90-секундното изчакване* (**Wait state**). Тази модификация опростила логиката и със сигурност намалила разходите, към което и са се стремели всички, участващи в разработката.

А ето и как всъщност се случва целият процес:



Фигура 5. The Work of Coca-Cola Vending Pass

Така изначалният успех на екипа на Кока-Кола провокирал разработиците да погледнат по-задълбочено върху **serverless computing** и, разбира се, да го имат предвид в бъдещи проекти. Благодарение на **Serverless Architecture** разработчиците повече няма да чакат, докато им бъдат осигурени сървъри, и ще могат свободно да разгърнат своята креативност и да преследват целите си. Те очакват да използват **Step Functions** за подобряване на мащабируемостта, функционалността и надеждността на своите приложения, с което значително да надхвърлят постигнатото чрез **Coca-Cola Vending Pass**. Пример за вижданията им след реализирането на **Coca-Cola Vending Pass** е намирането на „безсървърно решение“ за публикуване на хранителна информация за техните партньори за хранителни услуги чрез използването на Lambda Functions, Step Functions и API Gateway.

В заключение искам да добавя също и кои качествени характеристики на система, която е със Serverless Architecture, биват повлияни от избора на така реализирана архитектура:

Модулярност (Modularity): Система, чиято архитектура е serverless, има слаба модулна свързаност и зависимост (**low module coupling**), нейните компоненти са добре балансирани и са независими.

Преизползване (Reusability): Система, използваща serverless технологии, има малки единици код (units) и малки интерфейси.

Променяемост (Modifiability): Система със serverless архитектура следва да бъде лесно променяема, защото има прости единици код и слаба свързаност между отделните компоненти (coupling).

Тестваемост (Testability): Система, изоползваща serverless технологии, е лесно тестваема, защото има малко базов код, има прости единици код, състоящи се от независими компоненти.

Източници:

https://us.coca-cola.com/vending/how-it-works/

https://us.coca-cola.com/vending/

https://docs.aws.amazon.com/sns/latest/dg/welcome.html

https://aws.amazon.com/lambda/

https://aws.amazon.com/step-functions/

https://aws.amazon.com/blogs/aws/things-go-better-with-step-functions/

Изготвил:

Теодора Иванова

Фак.№ 62250, 1 група СИ