Technical Assessment

Agenda

Leader Election

Services

Limitations

Monitoring and Alerting

Testing

Leader Election

Pattern used in distributed computing

Each actor will act as a leader for a specific domain

“Fight” for a domain at start time and once it acquired the lock it will keep crawling

Send a healthcheck after each request to Zendesk to prove “it’s still working”

Services

The platform is made up of three services

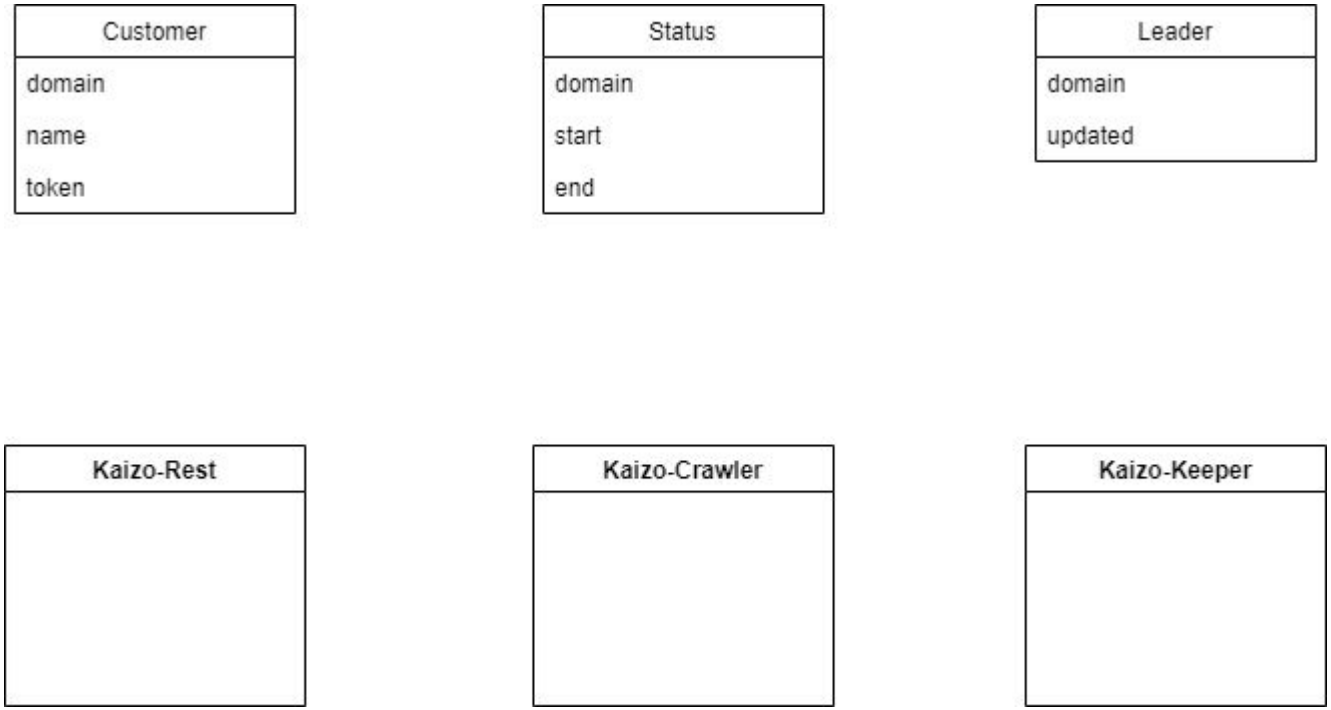
Kaizo Rest: a simple rest interface

Kaizo Crawler: the hard working rockstar of the platform which gets the data from zendesk

Kaizo Keeper: kind of a cron job, it makes sure that the platform will work accordingly after an unexpected incident

The data model is a very simple one, it only has three tables customer, status and leader

Services

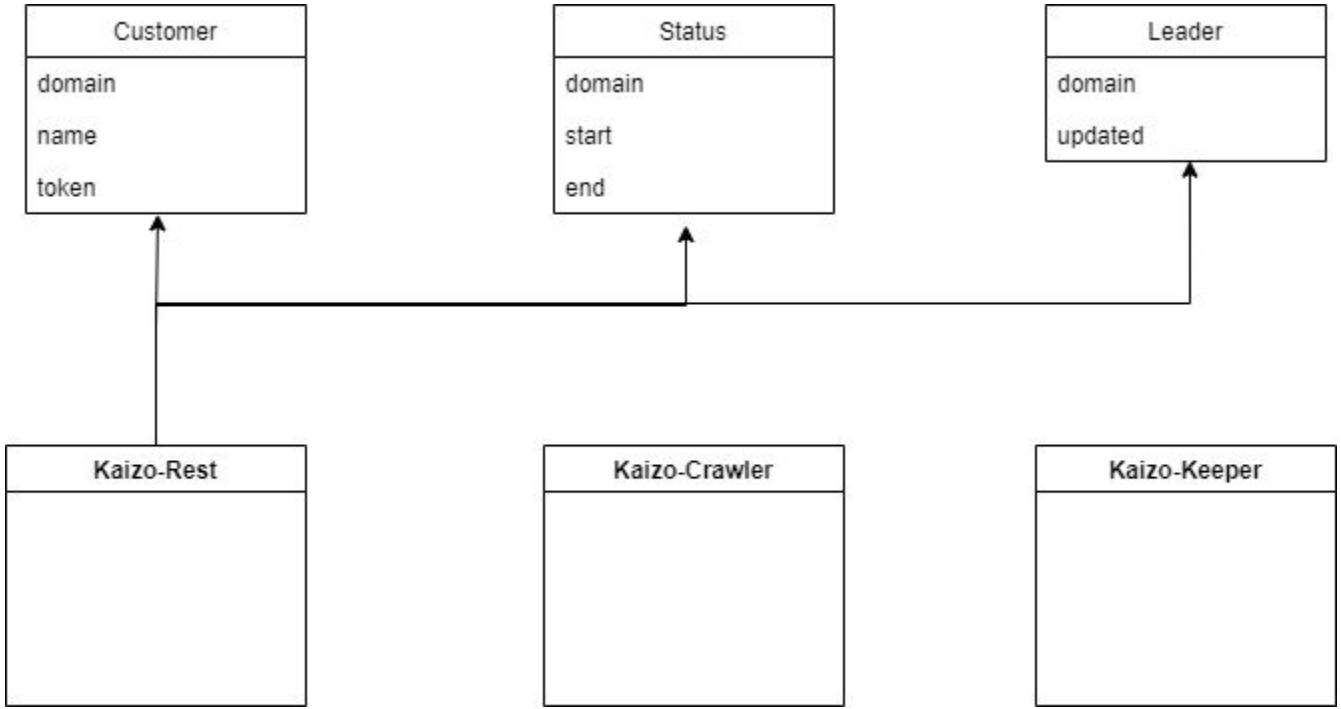

Customer
domain
name
token
Status
domain
start
end
Leader
domain
updated
Kaizo-Rest
Kaizo-Crawler
Kaizo-Keeper

Kaizo Rest

It is the main entry point for new domains in the platform

It also provides a way to check the status for each domain, to check when it was the last time it fetched data from ZenDesk

Kaizo Rest


Customer
domain
name
token
Status
domain
start
end
Leader
domain
updated
Kaizo-Rest
Kaizo-Crawler
Kaizo-Keeper

Kaizo Crawler

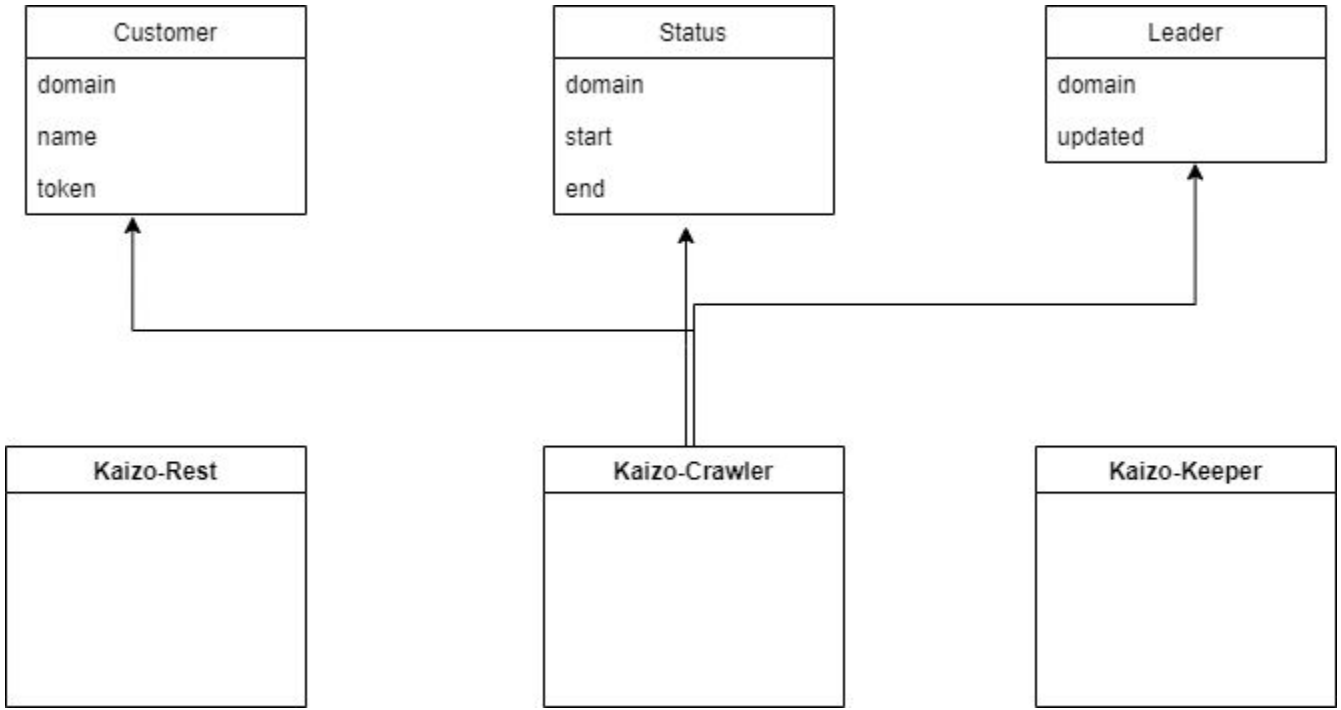
The “Rockstar” of the platform

It fetches all the unlocked domains and for each domain it creates a new actor which will try to lock that domain, if it can’t, then it will kill itself, otherwise it will keep crawling for tickets

The master actor periodically fetches records from Postgres looking for new domains available and repeats the process from above point

The worker actor periodically fetches tickets from Zendesk, it updates the status column which acts as an offset store and it also sends a healthcheck after each request

Kaizo Crawler



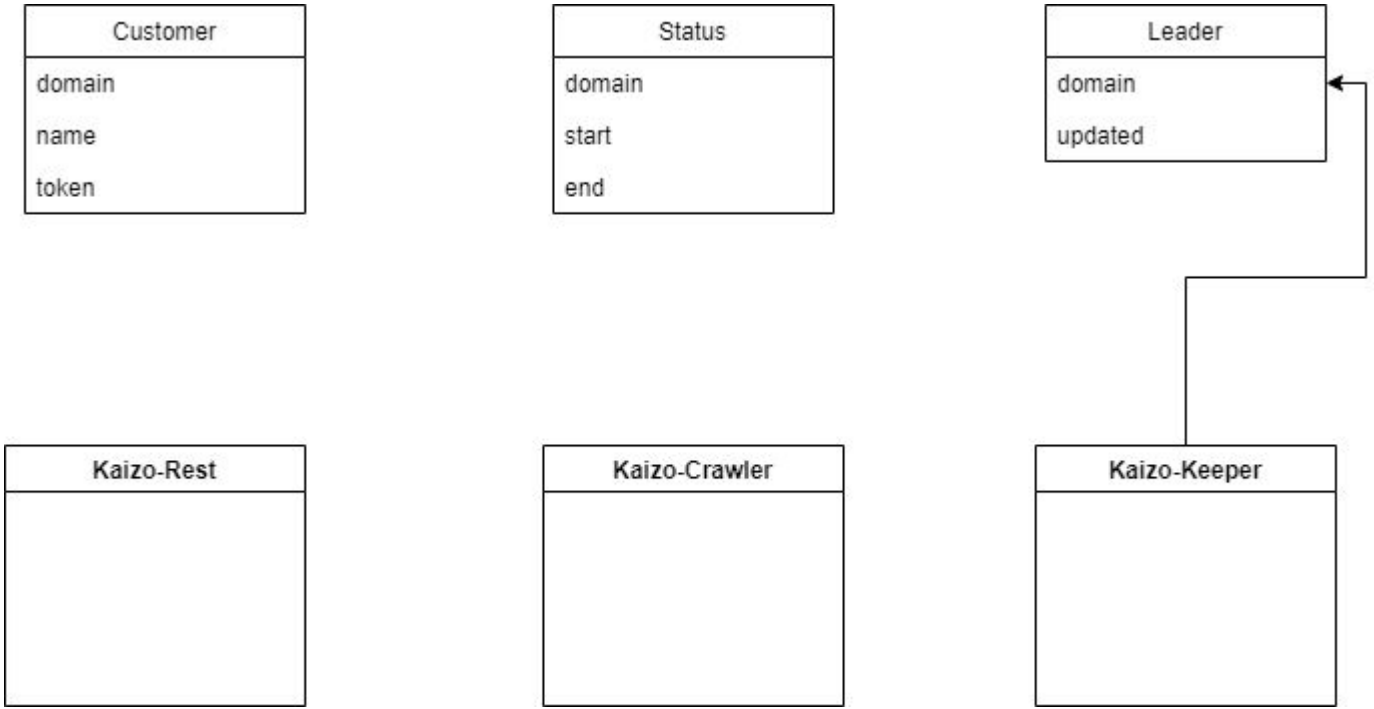
Kaizo Keeper

The “ZooKeeper” of the platform

It ensures that the platform will keep fetching data even if the actor in charge of a specific domain fails

If an actor hasn’t updated the leader table in a while, it will release that domain in order to be picked by new actors

Kaizo Keeper



Limitations

Delay for each new domain from the point in time when it was inserted using kaizo rest until the kaizo crawler checks the table for new entries

Delay for domains which were locked by actors who failed and in this case it might face some data loss issues

Monitoring and Alerting

Integration with a platform like DataDog to have access to real time metrics of the application, but also to have a system which alerts when the platform is below/above some thresholds, like the number of tickets ingested, the number of domain for which we fetch data

It is also good to understand if the application can be improved, if the resources are not fully used, if some parts can be split in smaller parts and executed in parallel

Testing

Create a proper testing environment

Mock server which acts as ZenDesk server with the same limitations for tickets and requests per minute

Perform different types of tests and also it can be tested how the platform will act in case the ZenDesk goes down for long period of time

Test different types of configuration, how the application will perform for a different number of instances, domains or when a lot of domains are added at once