KiwiCell your mobile cloud provider

Each second counts!

Kiwi Cell is a cloud based cellphone provider focused on crafting and launching new promotions based on their customers needs (facebook, specific user behavior, etc), and they are looking for a secure yet flexible platform to gain control of recharges and user consumption but also, allow the company to test and launch new campaigns in the shortest possible time.

Intended system user's

Personas

Peter

Peter - Line Owner

Recharges seller

Kiwi's employee

Our main character is Peter, he is our end user, and the one we need to provide perfect service, we also serve 2 other actors; each one of this interactions has its own considerations and definitions.

Peter wants to receive the best possible promotions based on what his is doing and what are his tastes.

Promotion design, testing and deployment has traditionally been one of the most complex tasks on Telco's commercial operation, having some cases where the process can take up to months.

Following we define the base guidelines and designs that defines how Kiwi's live time provisioning manager should be built and operated.

The main design drivers are **flexibility**, **security**, **reliability.**

We identify two core elements in the system, one is the user's facing information, and other the calculations and changes that can be made to the promotions structure.

In order to improve the capability to generate personalized promotions for users, we have opted for a 2 hybrid FIRE model, were external users get the latest information from Firebase, totally isolating them from the inner systems and improving our capability to consistently cache information, and "Reactions" are consumed and generated from Java backend to process all the business transactions and keep perfect synchrony among all actors.

Conceptual Model

Subscriber's live information and authentication

Transactional data, historics and auditing data

.Rest Facade
.Calculations
.Inner Security
.Promotions Manager

Kiwi's journey

The user journey maps Peter's life cycle, the direct and interactions he has with the other actors and base ideas to generate the best possible experience to Peter.

QENTEK amazing ux

Phase	Discover	Purchase	Provision	Consume
Description	Peter needs to make a phone call and discovers that he has no balance left	Hi uses his credit card on KiWi's app or goes to one of the airport quioscos to recharge his cellphone	The system applies the calculated promotion to peter and set the new balance into peter's account	Peter calls his friend, and the system alerts peter the next time he is running out of minutes
Service interactions	getMinsLeft	**getMinCost** **Discounts must be flexible, allowing to test and easily deploy promotions**	setRecharge setMinsLeft getMinsLeft	setMinutesUsed getMinsLeft

Components Model

Mysql

SuscriberData
fbRef
cellNr
used

Recharges
value
date
discount

Promotions
date
discount
module

Firebase is used to secure performance of end user facing data and total isolation from backend outages

Firebase

SuscriberData
discount
balance
used

GlobalData
basePrice

Rest Facade

Recharge
postRecharge

Mins
getLeft
getUsed
postUsed

Business Services

RechargeMgr
recharge

UsageMgr
setConsumedMins

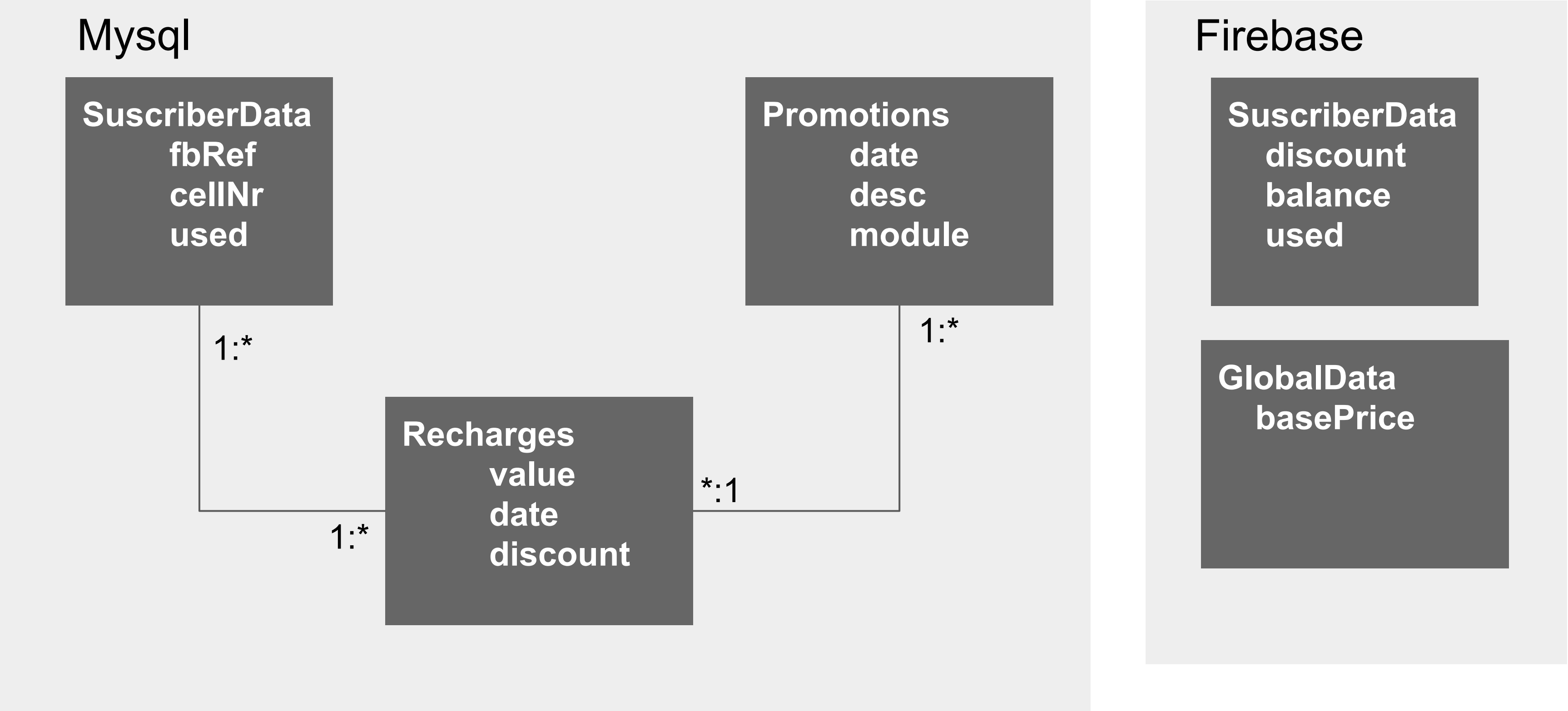
PromotionMgr
getDisccountVal
setCurrentPromotion

FB Reactors

Balance
usedMins

UsageMgr
setConsumedMins

DB Diagram



Mysql

SuscriberData
fbRef
cellNr
used

Promotions
date
desc
module

Recharges
value
date
discount

1:*

1:*

1:*

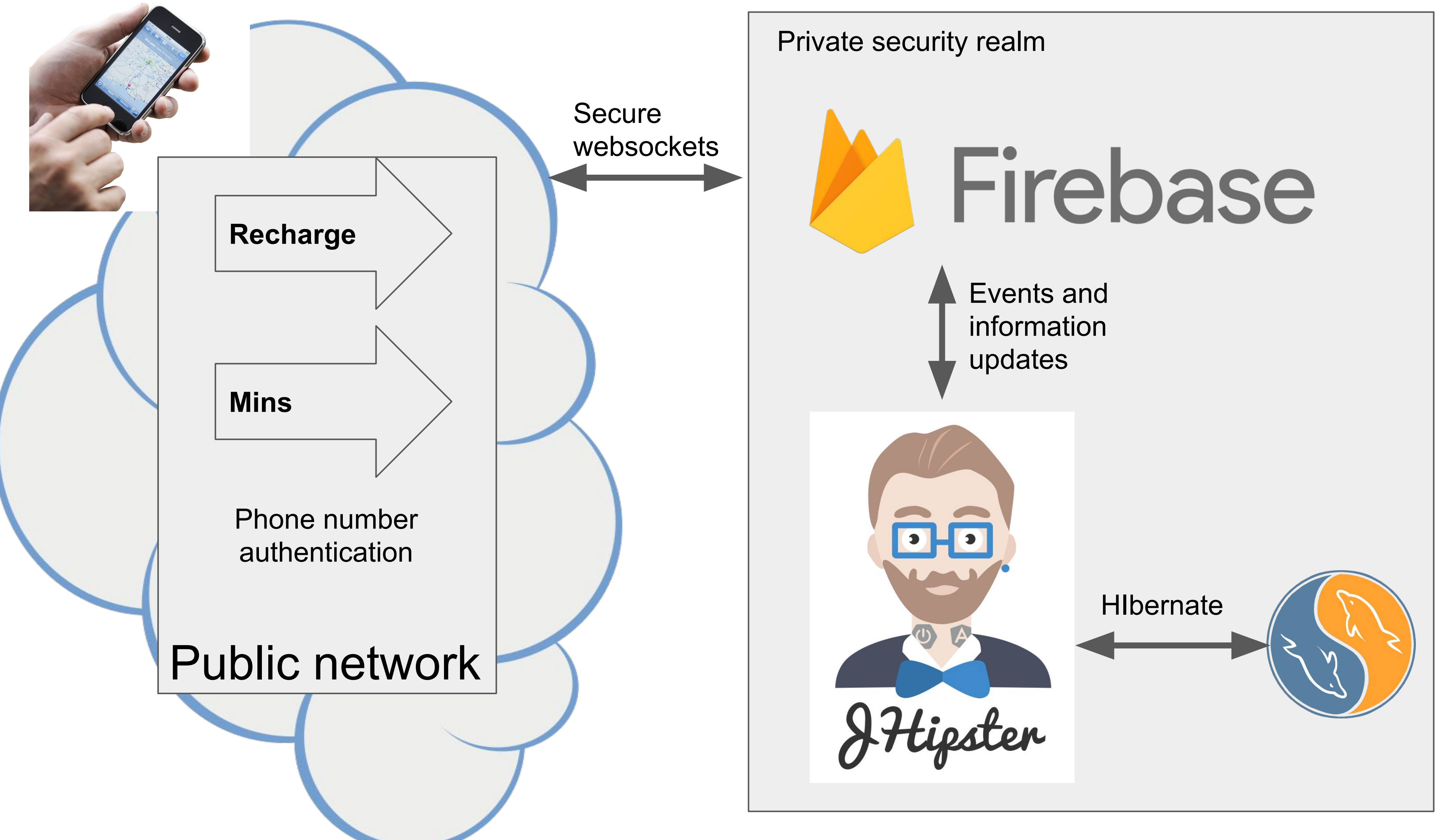
*:1

Firebase

SuscriberData
discount
balance
used

GlobalData
basePrice

Security&Interactions



Recharge

Mins

Phone number authentication

Public network

Secure websockets

Private security realm

Firebase

Events and information updates

JHipster

HIbernate

Recommendations

- For end users, HTTP communication can be resource consuming, websockets is recommended instead
- Data served through web services must be provided on singletons feed by firebase reactors in order to decrease the number of trips to the database
- For promotions, a business rules engine should be used instead of coding the calculations
- POC implementation is based on a simple monolithic application, but prod must be based on microservice architecture with a registry

Assumptions

- End users can switch communication channels
- Discount calculations won’t include the current recharge
- This system only gets the final report of seconds used by a client, controls to cut calls, provisioning other systems or communicating with third parties are outside the scope of the current design

Prepared by
Camilo Casadiego
28/09/2017