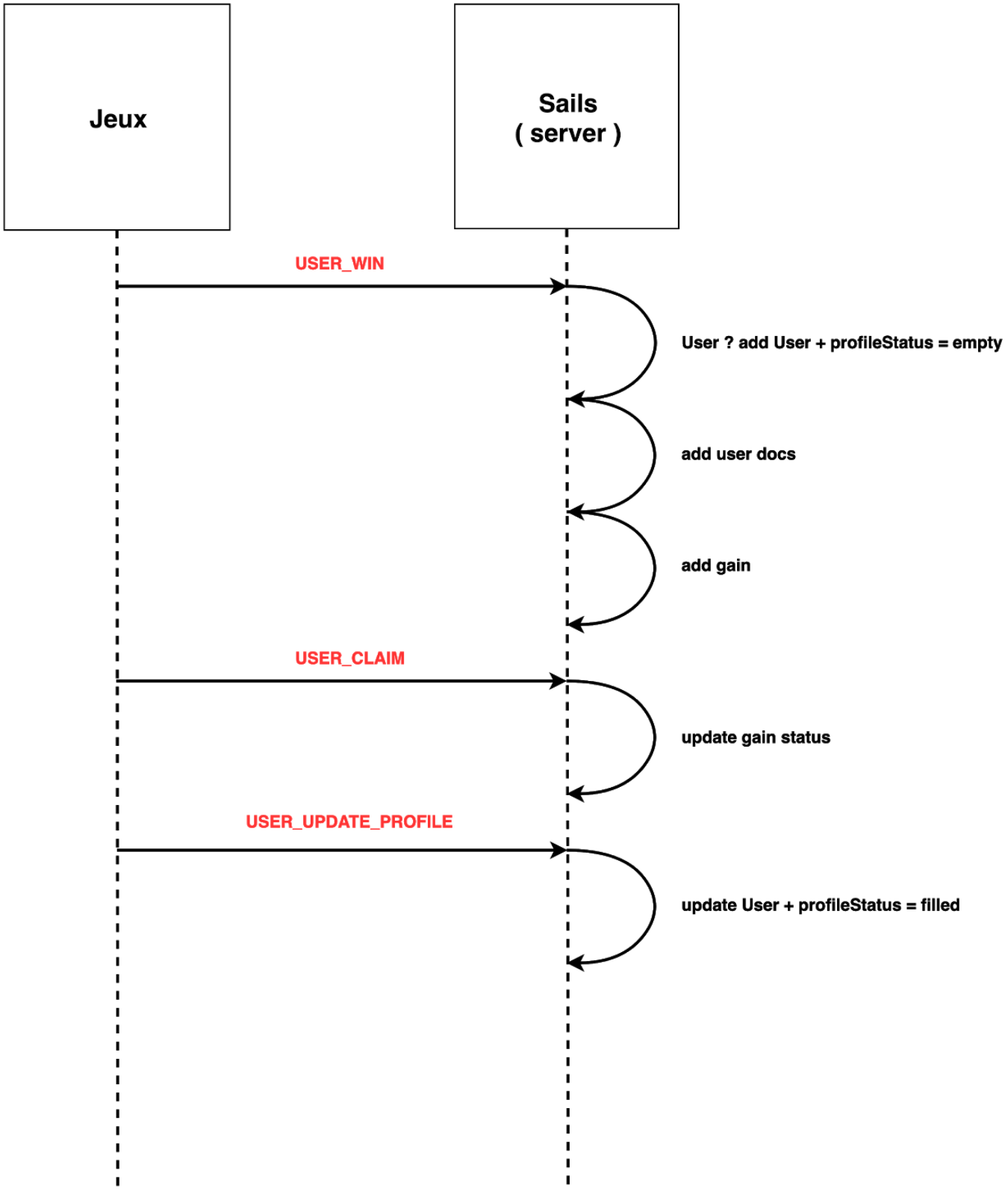
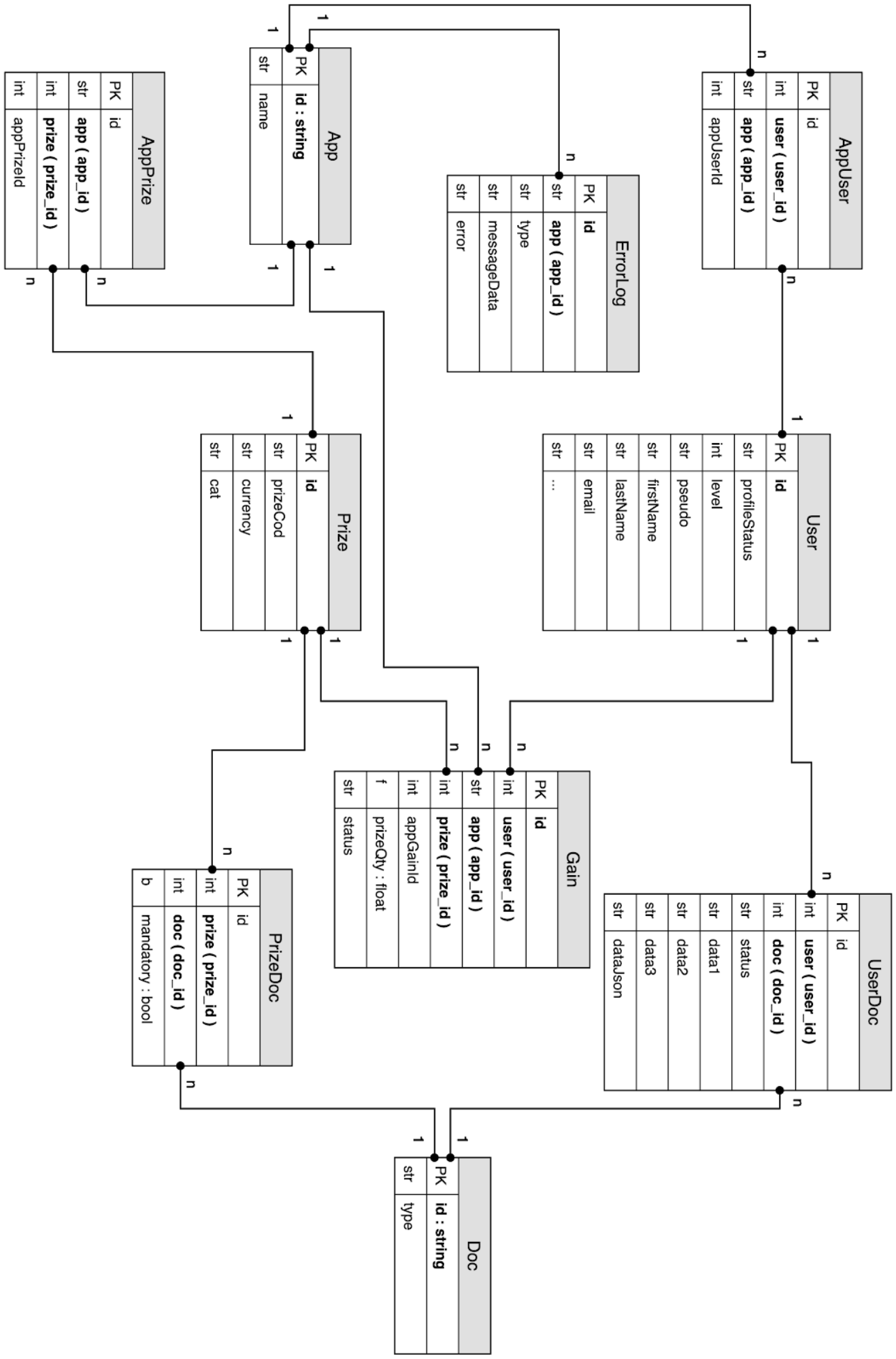
GAME USER SEQUENCE



Annexe 2


DB MODELS
AppUser
PK id
int user (user_id)
str app (app_id)
int appUserId
ErrorLog
PK id
str app (app_id)
str type
str messageData
str error
App
PK id : string
str name
AppPrize
PK id
str app (app_id)
int prize (prize_id)
int appPrizeId
User
PK id
str profileStatus
int level
str pseudo
str firstName
str lastName
str email
str ...
Prize
PK id
str prizeCod
str currency
str cat
Gain
PK id
int user (user_id)
str app (app_id)
int prize (prize_id)
int appGainId
f prizeQty : float
str status
PrizeDoc
PK id
int prize (prize_id)
int doc (doc_id)
b mandatory : bool
UserDoc
PK id
int user (user_id)
int doc (doc_id)
str status
str data1
str data2
str data3
str dataJson
Doc
PK id : string
str type

Annexe 3

Extrait de code 1

```
1  /**
2   * method to handle Game Event USER_WIN
3   * @param {} data data received from the sender, data sent should match : { event: USER_WIN, userId, appId, prizeId, prizeQty, status, gainId }
4   */
5  win: async function(data) {
6    // use transaction method to reset any modification in the DB if an error occur
7    await sails.getDatastore().transaction( async (db) => {
8      const gameUserObj = {
9        appUserId : data.userId,
10       app : data.appId
11     }
12
13     let existingAppUser = await AppUser.findOne(gameUserObj).usingConnection(db);
14
15     if ( typeof existingAppUser === 'undefined' ) {
16       const newUser = await User.create({}).fetch().usingConnection(db);
17
18       const newAppUser = await AppUser.create({ user: newUser.id, ...gameUserObj }).fetch().usingConnection(db);
19
20       existingAppUser = newAppUser;
21
22       const DocCollection = await Doc.find().usingConnection(db);
23
24       for(doc of DocCollection) {
25         await UserDoc.create({user: newUser.id, doc: doc.id, status: 'empty'}).usingConnection(db);
26       }
27     }
28
29     const gamePrizeObj = {
30       app: data.appId,
31       appPrizeId: data.prizeId
32     }
33
34     const existingAppPrize = await AppPrize.findOne(gamePrizeObj).usingConnection(db);
35
36     const gameGainObj = {
37       prizeQty: data.prizeQty,
38       status : data.status,
39       app: data.appId,
40       appGainId : data.gainId,
41     }
42
43     await Gain.findOrCreate(
44         { app: gameGainObj.app, appGainId: gameGainObj.appGainId },
45         { user: existingAppUser.user, prize: existingAppPrize.prize, ...gameGainObj }
46     )
47     .usingConnection(db);
48   })
49   // handle error however you would like, check sails documentation Response(`res`) & ORM/Errors
50   // note : here err is returned as a JSON, err.code is the error code ex: E_INVALID_NEW_RECORD, can also be a string for more general
     exception ex: 'failed' for connection issue
51   .intercept(err=> err);
52
53   return true;
54 }
55
```

Annexe 4

Extrait de code 2

```
1  /**
2   * method to handle Game Event USER_CLAIM
3   * @param {} data data received from the sender, data sent should match : { event: USER_CLAIMED, appId, gainId }
4   */
5  claim : async (data) => {
6  const existingGain = {
7      app : data.appId,
8      appGainId : data.gainId,
9  }
10
11 // import a function to define the new gain status, check ./fsm/
12 const nextGainStatus = require('../../../fsm/nextGainStatus');
13
14 const newStatus = await nextGainStatus(existingGain, 'USER_CLAIM');
15
16 // update the database with the status provided by the stateMachine
17 await Gain.updateOne(existingGain).set({status: newStatus}).intercept(err=>err);
18
19 return true;
20 }
21
```

Extrait de code 3

```
1  test('GameEventController.win is working with correct data', async () => {
2      const data = { event: 'USER_WIN', userId: 3, appId: 'BL', prizeId: 2, prizeQty: '3', status: 'init', gainId: 78 };
3
4      expect(await GameEventController.win(data)).toBe(true);
5  })
6
7  test('GameEventController.win create User', () => {
8      expect(User.created).toMatchObject([{ id: 1, profileStatus: 'empty' }]);
9  })
10
11 test('GameEventController.win create AppUser', () => {
12     expect(AppUser.created).toMatchObject([{ user: 1, appUserId: 3, app: 'BL' }]);
13 })
14
15 test('GameEventController.win create UserDoc', () => {
16     expect(UserDoc.created).toMatchObject([{ user: 1, doc: 1, status: 'empty' }, { user: 1, doc: 2, status: 'empty' }]);
17 })
18
19 test('GameEventController.win create Gain', () => {
20     expect(Gain.created).toMatchObject([{
21         user: 1,
22         prize: 24,
23         prizeQty: '3',
24         status: 'init',
25         appId: 'BL',
26         appGainId: 78
27     }]);
28 })
29
30
```

Annexe 5

Extrait de code 4

```
module.exports = AppUser = {
    created: [],

    findOne: function(searchParam) {
      const result = [...AppUser.created].filter(item => JSON.stringify(item) === JSON.stringify({...item, ...searchParam}) );
      const intercept = () => result[0];
      const usingConnection = () => result[0];
      return {intercept, usingConnection};
    ;},

    create: (object) => {
      AppUser.created.push(object);
      const intercept = () => object;
      const usingConnection = () => object;
      const fetch = () => {
        return {intercept, usingConnection};
      }
      return {fetch, intercept, usingConnection};
    }
  };


```

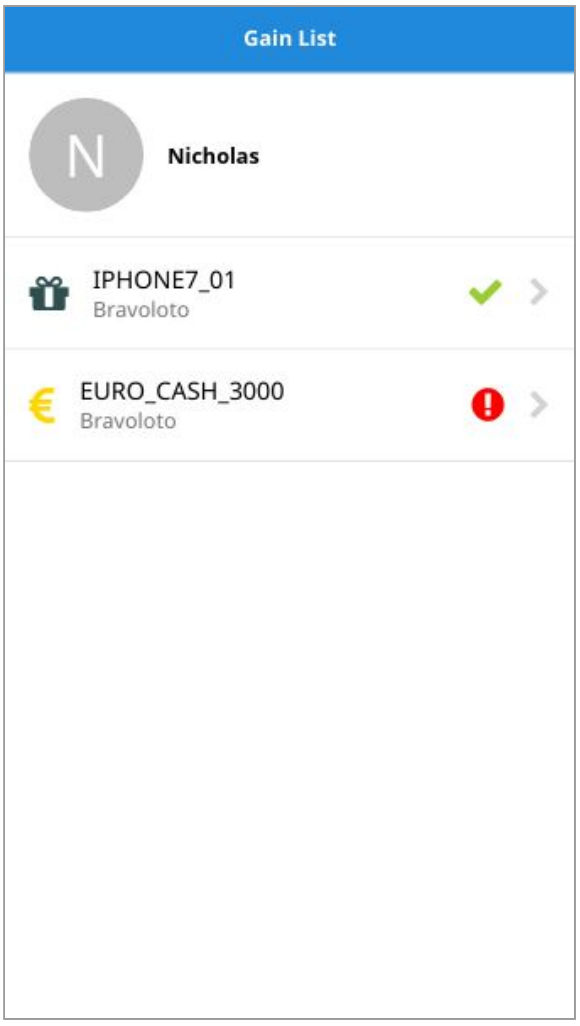
Extrait de code 5

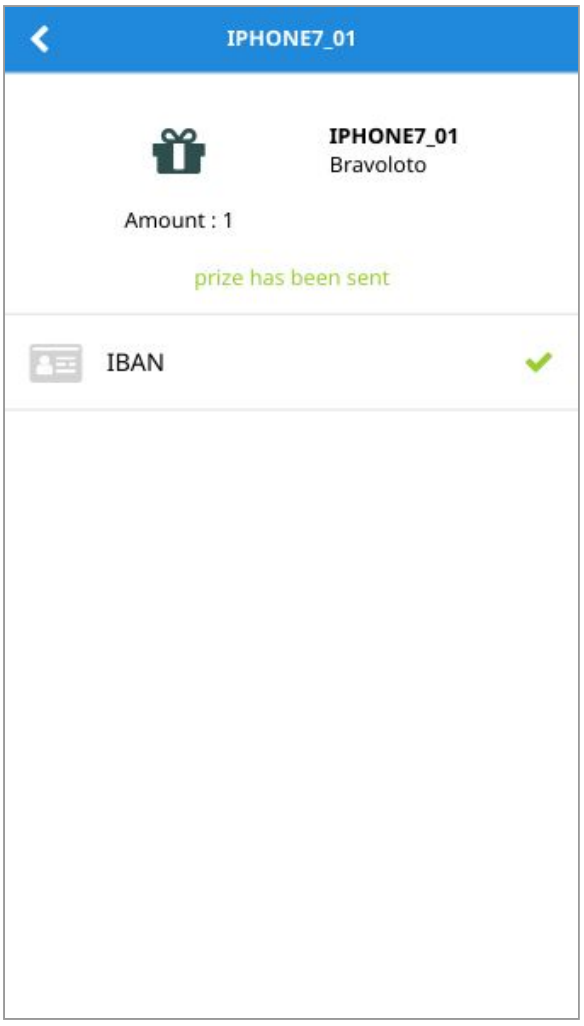
```
/**
 * called in receiver.js, handle restart if rabbitMQ's node is down based on a timer
 * @param {} qName name of the queue
 */
module.exports = handleReceiverError = (qName) => {
    // restart timer
    const restartTimer = 10000;
    // set worker name based on qName
    let workerName;
    switch (qName) {
      case 'EVENT_Q':
        workerName = workerEvent;
        break;
      case 'EVENT_ERROR_Q':
        workerName = workerError;
        break;
    }
    if (handleReceiverError[qName]) clearTimeout(handleReceiverError[qName]);
    handleReceiverError[qName] = setTimeout(workerName, restartTimer);
  }

```

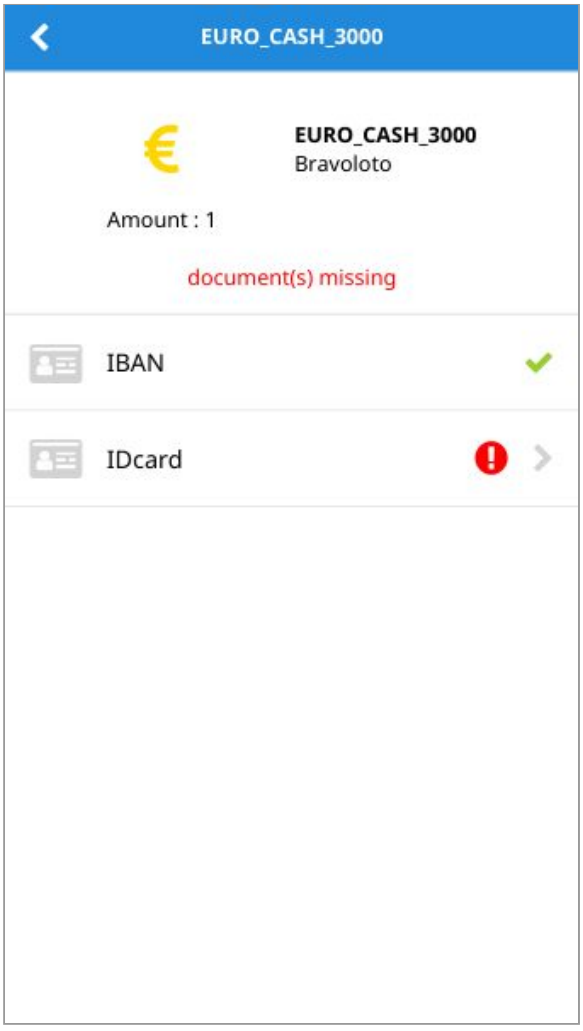
Extrait de code 6

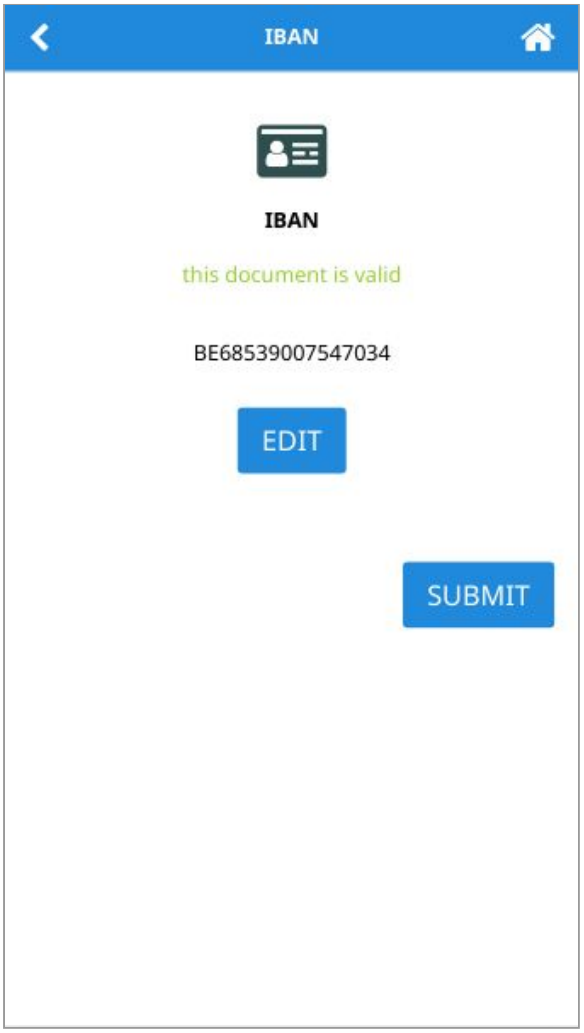
```
const receiver = require('../helpers/receiver');
const processError = require('../helpers/processError');
const processData = require('../helpers/processData');

module.exports = workerEvent = async () => await receiver('EVENT_Q', processData, processError, 'EVENT_ERROR_Q');


```

Annexe 6

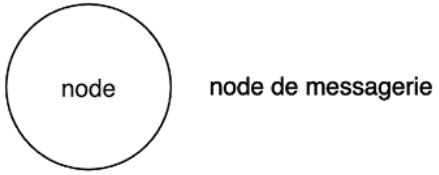

Gain List
N
Nicholas
IPHONE7_01
Bravoloto
EURO_CASH_3000
Bravoloto

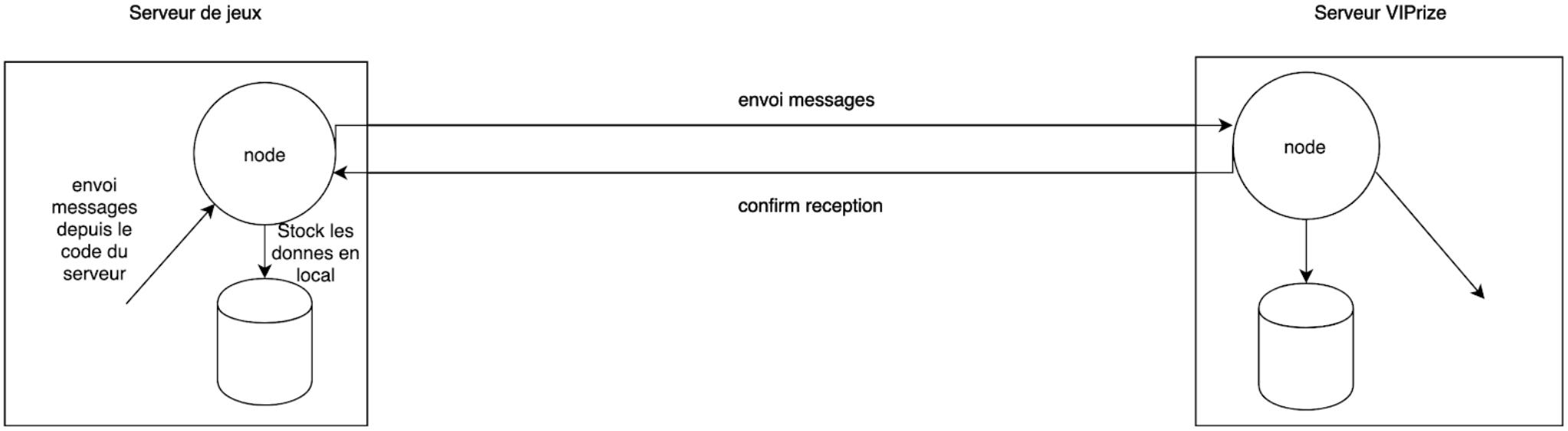

IPHONE7_01
IPHONE7_01
Bravoloto
Amount : 1
prize has been sent
IBAN


EURO_CASH_3000
EURO_CASH_3000
Bravoloto
Amount : 1
document(s) missing
IBAN
IDcard

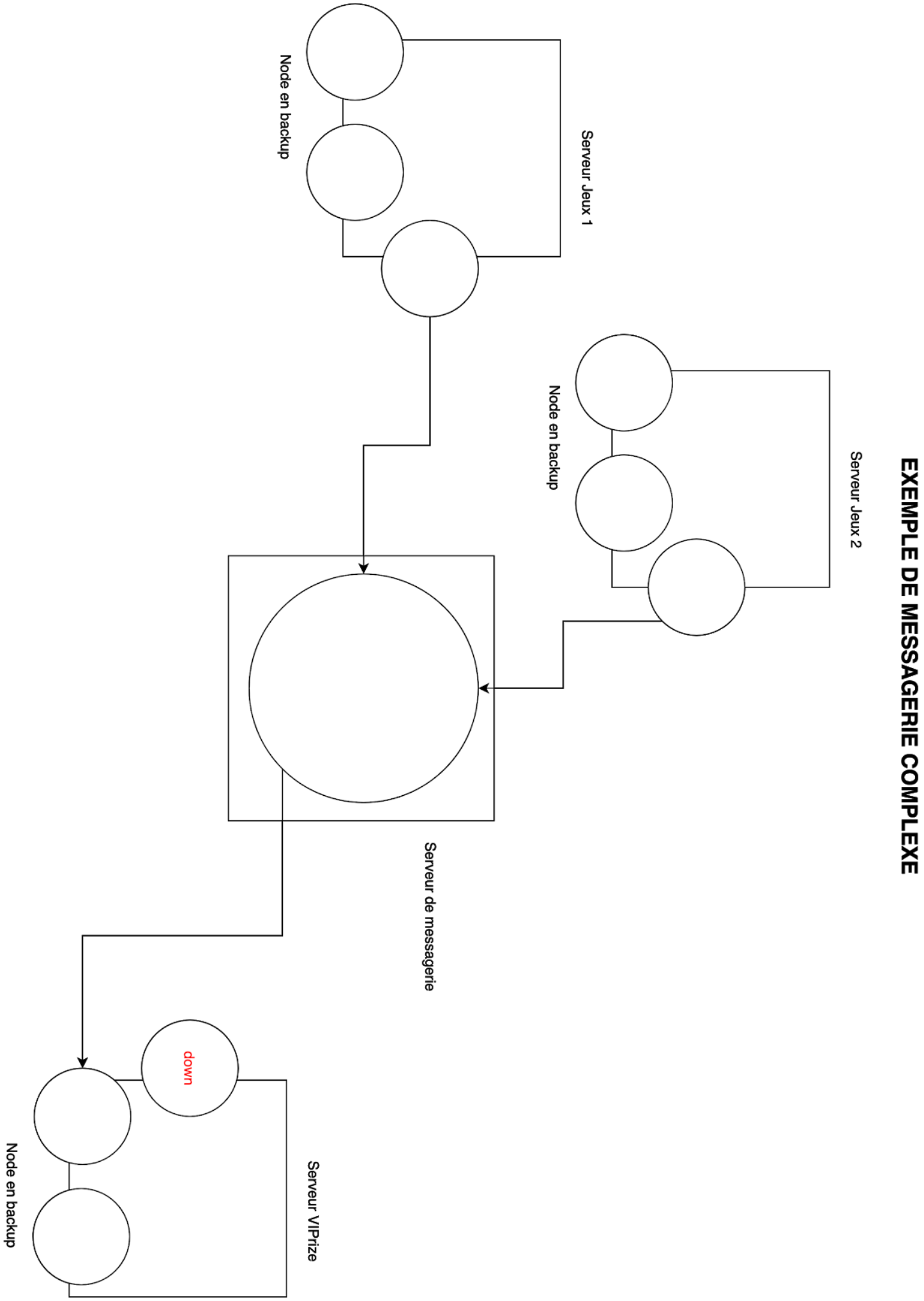

IBAN
IBAN
this document is valid
BE68539007547034
EDIT
SUBMIT

EXEMPLE SIMPLE DE MESSAGERIE





Annexe 8


EXEMPLE DE MESSAGERIE COMPLEXE
Serveur Jeux 2
Node en backup
Serveur Jeux 1
Node en backup
Serveur de messagerie
Serveur VIPrize
down
Node en backup