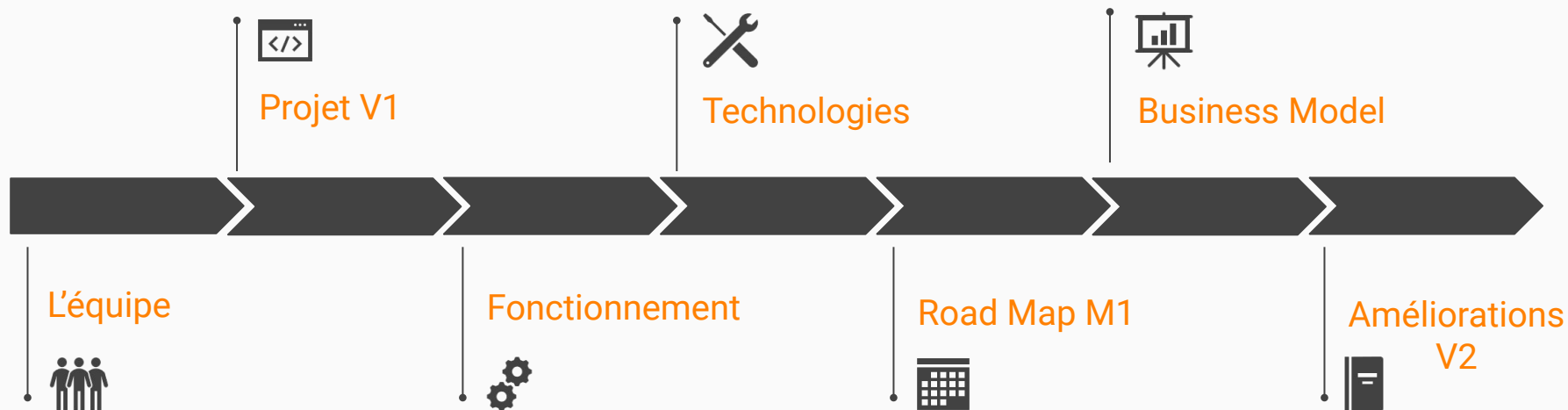




# Sommaire



# L'équipe



# L'équipe



**Paul Mussy**

Développeur  
GFI Informatique



[linkedin.com/paul-mussy](https://linkedin.com/paul-mussy)

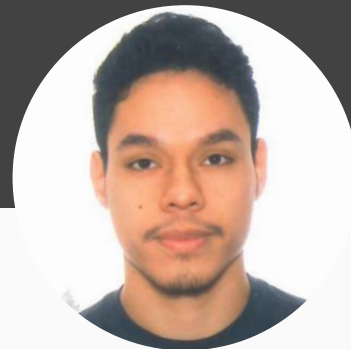


**Baptiste Louyot**

Développeur  
GFI Informatique



[linkedin.com/baptiste-louyot](https://linkedin.com/baptiste-louyot)



**Juan Jose Urrego**

Développeur  
Finelia



[linkedin.com/juan-jose-urrego](https://linkedin.com/juan-jose-urrego)





# Projet V1.





# Projet

Application Décentralisée pour location d'objets.

Location d'objets ou de biens automatisés à l'aide d'une interface utilisateur (DApp) couplée avec un appareil connectés contrôlé par un smart contract.



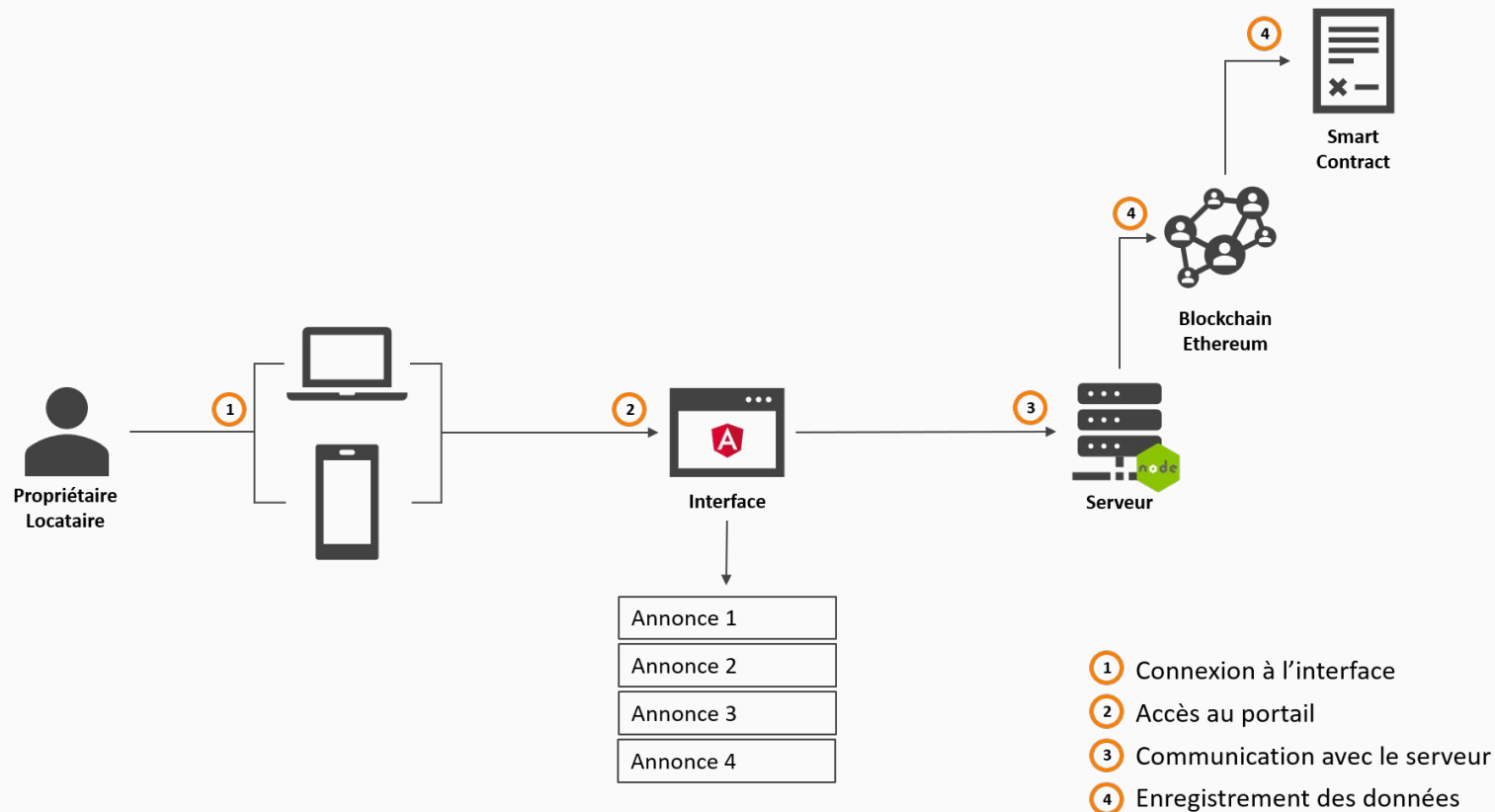
# Fonctionnement







# Fonctionnement 1/2







## Fonctionnement 2/2

### Propriétaire

Dans la logique de Airbnb, il décide de vouloir louer son objet/bien, il va se rendre sur l'interface utilisateur pour créer le contrat de location à l'aide d'un formulaire. Afin de terminer le processus il faudra coupler l'objet ou le bien avec un système connecté.

### Locataire

Pour le locataire dès qu'il a besoin d'un objet il lui faut se rendre sur le market de l'interface et de louer l'objet qu'il souhaite. Il doit ensuite payer la somme de location ainsi que la caution avec notre « stable coin ». Une fois le paiement effectué l'objet pourra se déverrouiller avec l'ID du locataire et le système connecté.

### Système

Le smart contract remplira la fonction de l'employé de la société de location. Création, modification et suppression du contrat de location. Gestion utilisateur sur l'interface web liée au smart contract pour les infos.

# Technologies





# Objectifs technologiques

- Réalisation d'une interface web utilisateur.
- Création d'un smart contract à l'aide de solidity.
- Création d'un système connecté.



# Interface Web

## Outils technologiques

### FRONT

- NodeJS + Express
  - EJS
- MBootstrap
- HTML / CSS

### BACK

- NodeJS + Express
- Mysql

## Front-end

NodeJS + Express  
MBootstrap

## Back-end

NodeJS + Express  
Mysql





## Connexion

Email (\*) :

Mot de passe (\*) :

☐ Se souvenir de moi    [Mot de passe oublié ?](#)

SE CONNECTER

Pas encore inscrit ? [S'inscrire](#)

## Inscription

Prénom (\*) :

Nom (\*) :

baptiste.louyot@gmail.com

....

Au moins 8 caractères

Confirmation (\*) :

S'INSCRIRE

En cliquant sur *S'inscrire* tu es d'accord avec  
[les termes de service](#)

## Connexion Inscription

L'interface est intégralement visible à l'utilisateur non connecté, mais aucune interaction n'est possible.

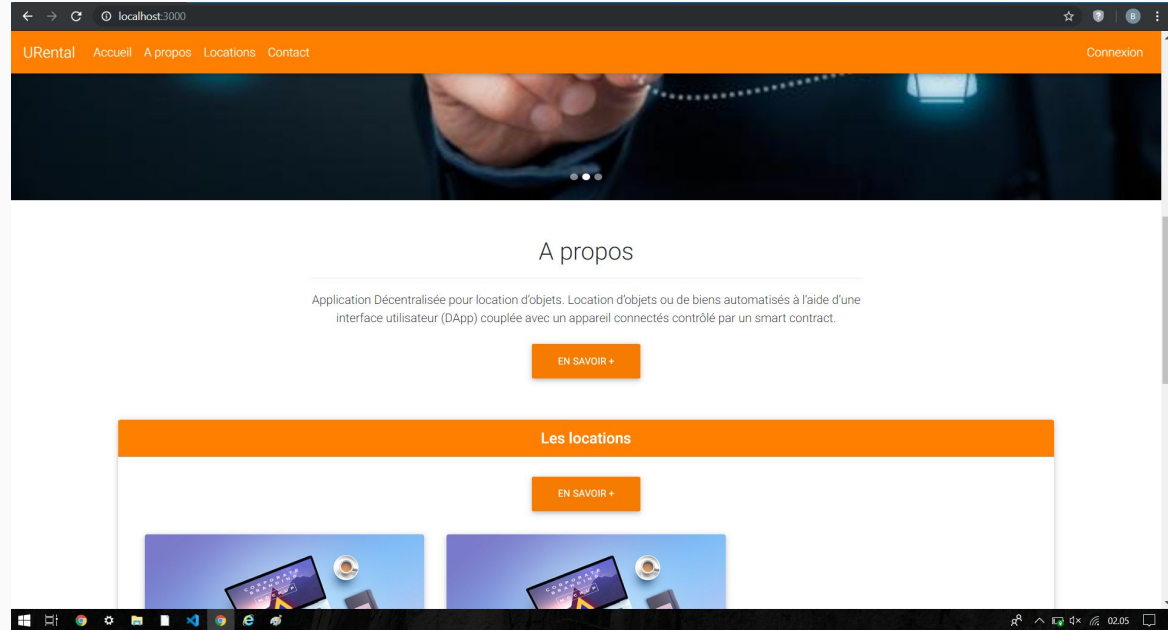
Un compte permet d'accéder à toutes les fonctionnalités de l'interface.

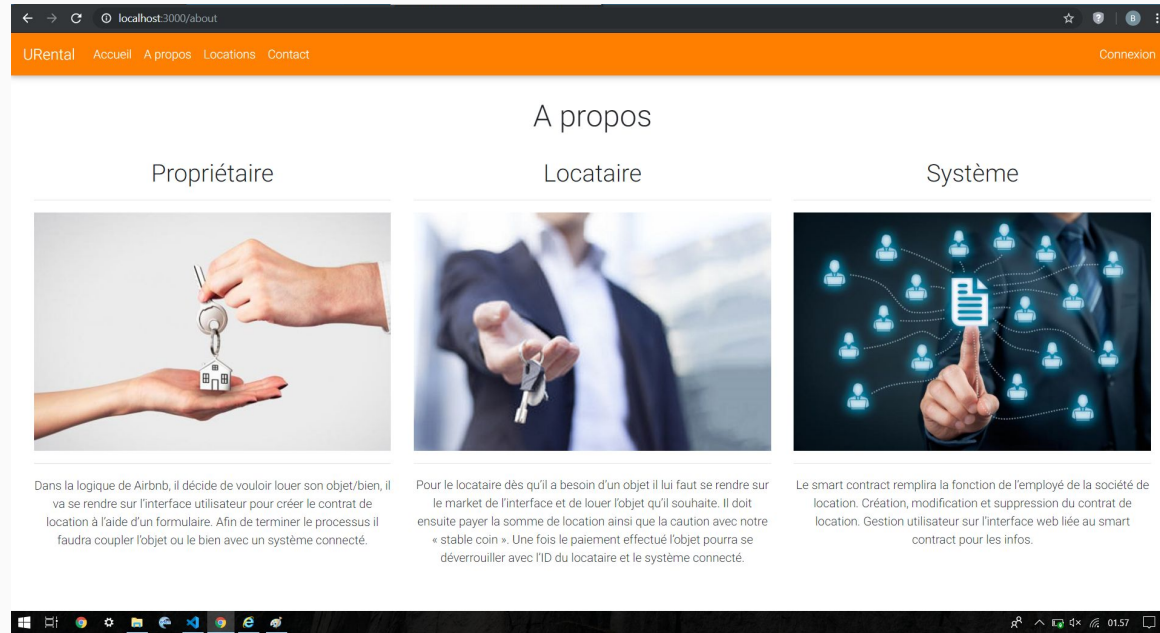


## Accueil

Présentation de  
l'ensemble des  
fonctionnalités et  
contenu présent sur  
l'interface web.

Une carte de visite avec  
les 3 derniers biens  
ajoutés.





## A propos

### Explication du projet

#### Point de vue :

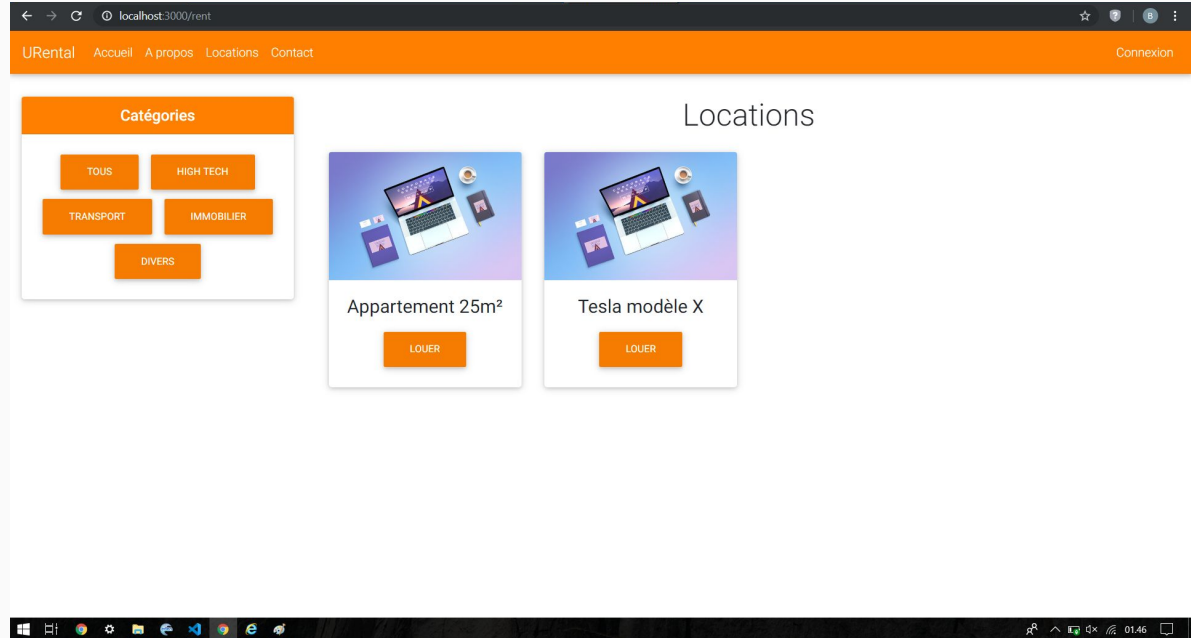
- Propriétaire
- Locataire
- Système



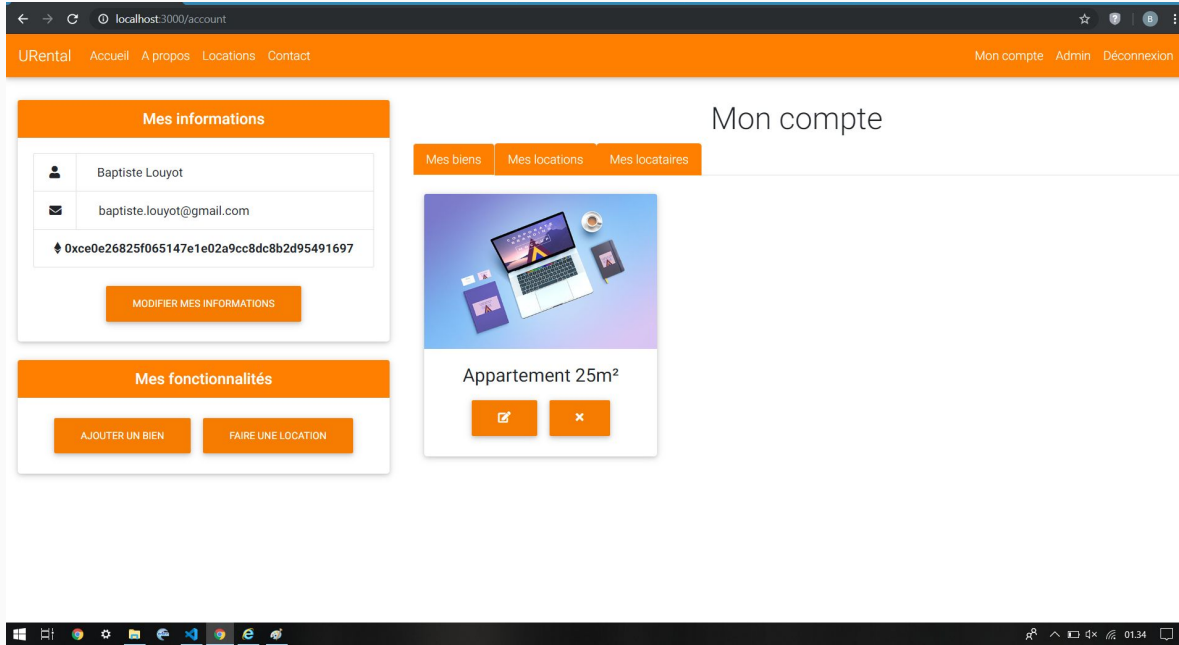
## Locations

Une vitrine de tous les biens mise en ligne sur l'interface.

La possibilité en un clic d'avoir toutes les informations du bien en question ainsi qu'un formulaire de location pour effectuer une location.







## Mon compte

Aperçu de toutes les informations de ses données.

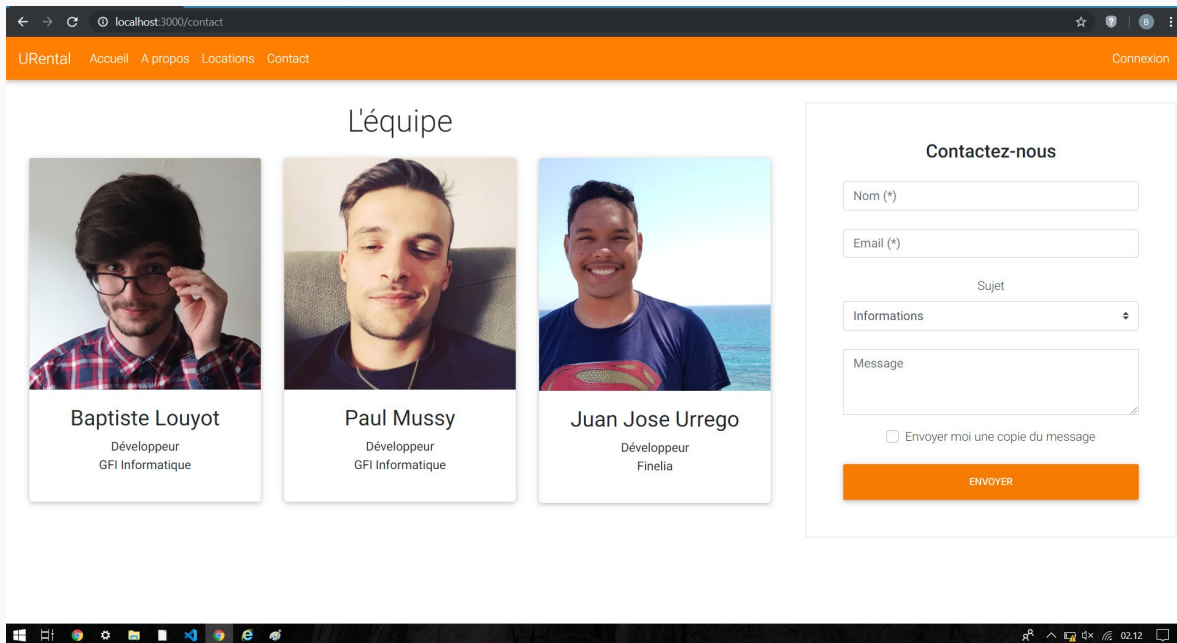
Gestion de ses biens et locations.

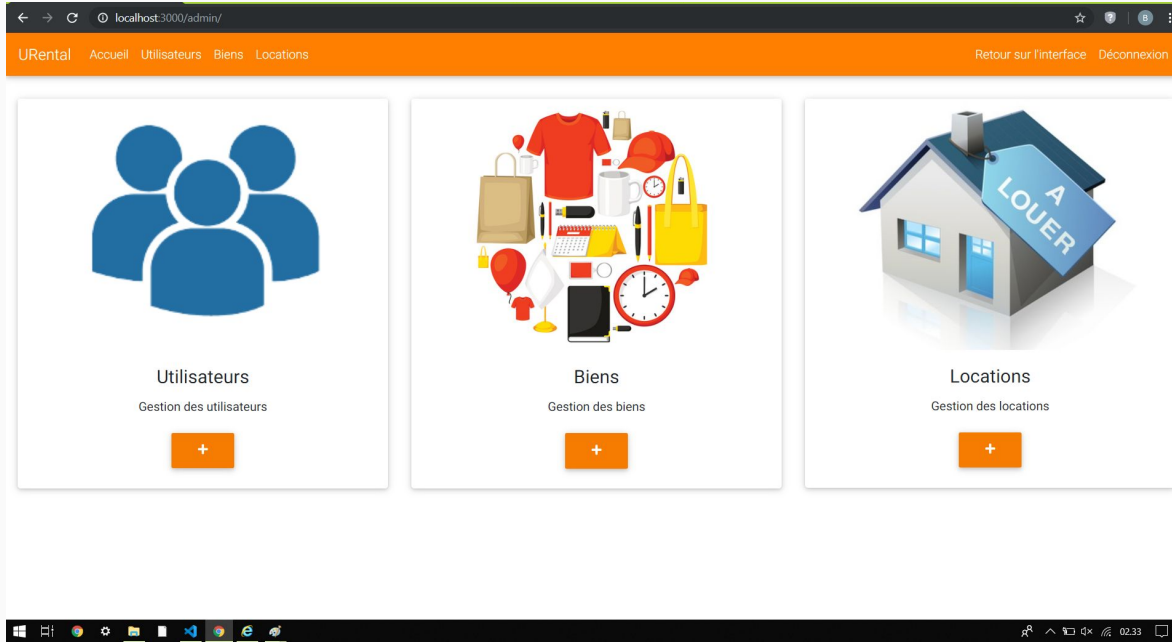
- Ajout
- Modification
- Suppression



## Contact

Présentation de l'équipe  
avec un formulaire de  
contact pour pouvoir  
poser toutes les questions  
concernant le projet.





## Accueil

Tableau de bord de la partie admin.

Cette session permet de voir l'ensemble des gestions



## Utilisateurs

Gestion de tous les utilisateurs de l'interface.

#	Prénom	Nom	Email	Action
1	Baptiste	Louyot	baptiste.louyot@gmail.com	
2	Paul	Mussy	paul.mussy@gmail.com	
3	Juan Jose	Urrego	juan-jose.urrego@gmail.com	















localhost:3000/admin/items

URental Accueil Utilisateurs Biens Locations Retour sur l'interface Déconnexion

### Catégories

#	Nom	Action
1	High tech	 
2	Transport	 
3	Immobilier	 
4	Divers	 

### Biens

#	Nom	Description	Prix / Format	Cauton	Action
1	Appartement 25m²	Appartement de 25m². Situé 15 rue saint paul, 75004 Paris	0.5ETH / jour	0.3ETH	
2	Tesla modèle X	Voiture électrique	0.5ETH / jour	1ETH	

## Biens

Gestion des biens de tous les utilisateurs.

Gestion des catégories.



## Locations

Affichage des locations de  
tous les utilisateurs.

#	Id item	Id renter	Date de début	Période
1	2	1	Sun Jul 28 2019 00:00:00 GMT+0200 (GMT+02:00)	86400



# Interface Web - App.js

## Code source du fichier App.js

```
JS app.js > getModifyItemPage
64   cookie: { secure: false }
65   )))
66
67   // CONFIG
68   app.set('port', port); // set express to use this port
69   app.set('views', __dirname + '/views'); // set express to look in this folder to render our view
70   app.set('view engine', 'ejs'); // configure template engine
71   app.use(bodyParser.urlencoded({ extended: false }));
72   app.use(bodyParser.json()); // parse form data client
73   app.use(express.static(path.join(__dirname, '/'))); // configure express to use public folder
74
75   // GET
76   app.get('/', getIndexPage);
77   app.get('/about', getAboutPage);
78   app.get('/rent', getRentPage);
79   app.get('/rent/category/:id', getRentCategoryPage);
80   app.get('/delete-rent/:id', getDeleteRent);
81   app.get('/item/:id', getItemPage);
82   app.get('/add-item', addItemPage);
83   app.get('/modify-item/:id', getModifyItemPage);
84   app.get('/delete-item/:id', getDeleteItem);
85   app.get('/contact', getContactPage);
86   app.get('/account', getAccountPage);
87   app.get('/modify-account', getModifyAccountPage);
88   app.get('/subscribe', getSubscribePage);
89   app.get('/login', getLoginPage);
90   app.get('/logout', getLogout);
91
```

```
JS app.js
39   getAdminDeleteItem,
40   getAdminDeleteUser,
41   getAdminDeleteRent} = require('./routes/admin');
42
43   // DATABASE
44   const db = mysql.createConnection ({
45     host: 'localhost',
46     user: 'root',
47     password: '',
48     database: 'urental'
49   });
50
51   db.connect((err) => {
52     if (err) {
53       throw err;
54     }
55     console.log('Connected to database');
56   });
57   global.db = db;
58
59   // VARIABLE SESSION
60   app.use(session({
61     secret: 'urental',
62     resave: false,
63     saveUninitialized: false,
64     cookie: { secure: false }
65   }));
66
```

# Développement Solidity

- DApp
  - Web3
  - NodeJS + Express
- Smart Contract
  - Solidity
- Environnement de test
  - Geth
  - Truffle



**Liaison avec l'interface**  
Web3.js + NodeJS + Express



**Blockchain Ethereum  
Testnet**

Solidity + Geth + Truffle





# Web3.js - App.js

## Web3.js v 1.0

Initialisation de Web3 & des  
variable du contrat

```
// WEB3 interface
const abi = require("../assets/js/contractABI");
let web3 = new Web3(new Web3.providers.HttpProvider("http://localhost:8575"));
let urenalContract = new web3.eth.Contract(abi.abi, '0x00838625d3BAf935a4ac15c7a945b1483472059c');
let ownerAccountAddress = '0xce0e26825f065147e1e02a9cc8dc8b2d95491697';
let buyerAddress = "0x206e9d4314ea576f1a735b72e101ce8f82b0d222";
let sellerAddress = "0xd3df5996466965a2efe35b84e9efce909d5c3294";
```

```
db.query(userQuery, (err, resultUser) => {
  if (err) return res.status(500).send(err);
  req.body.userSeller = resultUser[0];
  let finalPrice = parseFloat(req.body.caution) + (parseFloat(req.body.price) * parseFloat(rent.period));
  let hash = web3.utils.keccak256(result.insertId + "" + req.body.userSeller.address_etherium + "" + req.body.caution + "" + req.body.newPeriod + "" + req.body.userBuyer.ad);
  let hexHash = web3.utils.toHex(hash);
  urenalContract.methods.rent(hexHash, req.body.userSeller.address_etherium, web3.utils.toWei(req.body.caution.toString(), "ether"), req.body.newPeriod).send({
    from: req.body.userBuyer.address_etherium,
    gasPrice: web3.utils.toHex(20000000000),
    gasLimit: web3.utils.toHex(4700000),
    value: web3.utils.toWei(finalPrice.toString(), "ether")
  })
  .on('error', function (error) {
    console.log("error : ");
    console.log(error);
  })
  .on('transactionHash', function (transactionHash) {
  })
  .on('receipt', function (receipt) {
    console.log("receipt : ");
    console.log(receipt);
  })
  .on('confirmation', function (confirmationNumber, receipt) {
    console.log("confirmation");
    console.log(receipt);
    console.log(confirmationNumber);
    let hash = web3.utils.keccak256(result.insertId + "" + req.body.userSeller.address_etherium + "" + req.body.caution + "" + req.body.newPeriod + "" + req.body.user);
    let hashQuery = "UPDATE rents SET hash = '" + hash + "' WHERE id = " + result.insertId + """;
    db.query(hashQuery, (err, result) => {
      if (err) return res.status(500).send(err);
      res.redirect('/account');
    });
  });
});
});
```

Appel de la fonction  
principale rent, qui envoie les  
infos au Smart Contract



## Appel des fonctions:

- Fonction launchRent:  
lance le compteur qui  
permet de calculer la  
durée de location
- Fonction releaseRent:  
accessible une fois la  
période de location  
terminée, déclenche les  
paiements en fonctions  
des states des parties  
prenantes.

```
app.route('/launchRent').post(function(req,res){
  uralContract.methods.launchRent( web3.utils.toHex(req.body.rentHash) ).send({
    from: req.session.addressEthereum,
    gasPrice: web3.utils.toHex(20000000000),
    gasLimit: web3.utils.toHex(4700000) })
    .on('error', function(error){
      console.log( "error : ");
      console.log( error )
    })
    .on('transactionHash', function(transactionHash){})
    .on('receipt', function(receipt){
      console.log("receipt : ");
      console.log(receipt)
    })
    .on('confirmation', function(confirmationNumber, receipt){
      console.log("confirmation");
      console.log(receipt);
      console.log(confirmationNumber);
      res.redirect('/account');
    });
});

app.route('/releaseRent').post(function(req,res){
  uralContract.methods.releaseRent( web3.utils.toHex(req.body.rentHash) ).send({
    from: req.session.addressEthereum,
    gasPrice: web3.utils.toHex(20000000000),
    gasLimit: web3.utils.toHex(4700000) })
    .on('error', function(error){
      console.log( "error : ");
      console.log( error )
    })
    .on('transactionHash', function(transactionHash){})
    .on('receipt', function(receipt){
      console.log("receipt : ");
      console.log(receipt)
    })
    .on('confirmation', function(confirmationNumber, receipt){
      console.log("confirmation");
      console.log(receipt);
      console.log(confirmationNumber);
      res.redirect('/account');
    });
});
```



# Smart Contract

```
contract urentalRents is Ownable{

    uint8 constant ACTION_BUYER_RENT = 0x01;
    uint8 constant ACTION_BUYER_START_RENT = 0x02;
    uint8 constant ACTION_BUYER_RELEASE_RENT = 0x03;
    uint8 constant ACTION_BUYER_CANCEL_RENT = 0x04;
    uint8 constant ACTION_SELLER_ACCEPT_RENT = 0x05;
    uint8 constant ACTION_SELLER_REQUEST_CAUTION = 0x06;
    uint8 constant ACTION_SELLER_ACCEPT_RELEASE = 0x07;
    uint8 constant ACTION_RENT_BLOCKED = 0x08;
    uint8 constant ACTION_RENT_IN_USE = 0x09;
    uint8 constant ACTION_RENT_FINISHED = 0x10;

    struct Rent {
        address payable buyer;
        address payable seller;
        uint256 amount;
        uint256 caution;
        uint256 duration;
        uint8 stateBuyer;
        uint8 stateSeller;
        uint8 stateRent;
        uint start;
    }

    mapping(address => uint256) balances;
    mapping(bytes32 => Rent) private rents;

    modifier onlyAfter(bytes32 _id) {[_id]}

    modifier onlyBuyer (bytes32 _id) {[_id]}
    modifier onlySeller (bytes32 _id) {[_id]}

    modifier onlyRentInUse(bytes32 _id) {[_id]}
    modifier rentExist(bytes32 _id) {[_id]}
```

Constante: Sert à gérer les différents états des parties prenantes et de la location en cours.

Structure Rent: Stock l'intégralité des paramètres de la location dans un mapping.

Modifier: Gère l'accès aux différentes fonctions et la gestion du temps de location.

Structure, modifier, constante de gestion



- Solidity 0.5.0

## Prototypes de fonctions



# Mise en place Geth & Truffle

## Lancement de Geth

```
ubuntu@ip-172-31-87-153:~$ geth --testnet --syncmode "light" --networkid 3 --rpc --  
rpcapi "eth,net,web3" --rpccorsdomain '*' --rpcaddr 127.0.0.1 --rpcport 8575 --allo  
w-insecure-unlock --unlock "0,1,2" --password "pwd.txt" console
```

## Compilation et déploiement avec truffle

```
ubuntu@ip-172-31-87-153:~/truffle/HelloWorld$  
ubuntu@ip-172-31-87-153:~/truffle/HelloWorld$ truffle init  
ubuntu@ip-172-31-87-153:~/truffle/HelloWorld$  
ubuntu@ip-172-31-87-153:~/truffle/HelloWorld$ truffle compile  
ubuntu@ip-172-31-87-153:~/truffle/HelloWorld$ truffle migrate
```

[Prototypes de fonctions](#)



# Business Model



# Road map

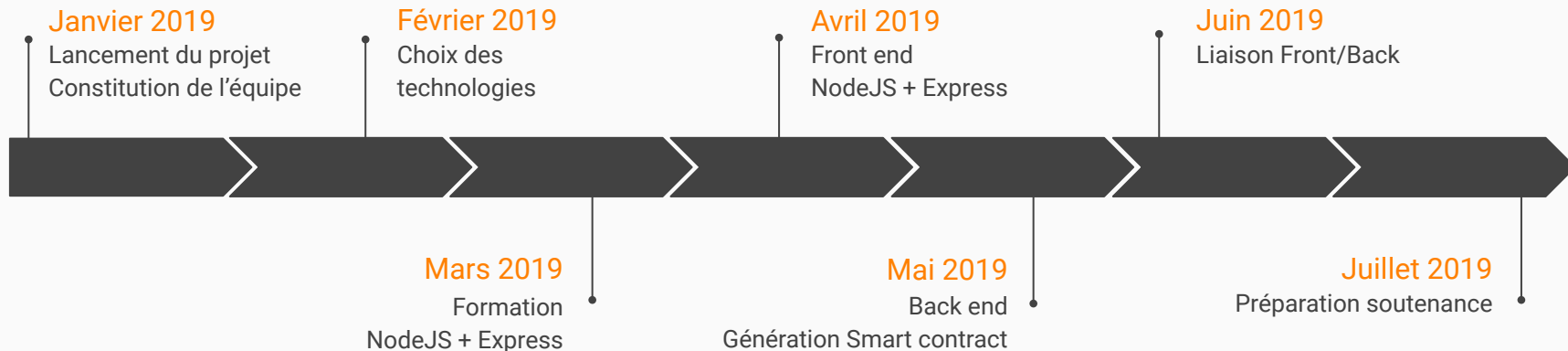






# Road Map

*Master 1*



La réalisation du système connecté et ça liaison à notre App concerne le Master 2.

# Améliorations V2.





## Système connecté

- Mise en place d'un système connecté, ou liaison avec un système existant (type digicode)

## Interface

- Gestion des images pour les biens
- Gestion des locations améliorés
  - Aperçu plus détaillé des informations de la location
  - Gestion du

## Smart Contract

- Pouvoir cancel la réservation
- Prendre la gestion d'événement en compte

## Web3.js

- Gestion des événements renvoyés par le smart contract
- Liaison avec Metamask (échec cette année)

# Démo !

