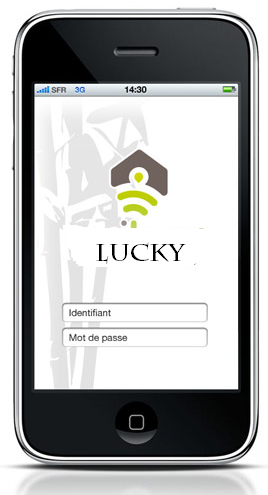
(/!\ Concernant les images, ce sont des captures d’écran qui ont été réalisées. Ne pas tenir compte des images et de certaines choses qui ne doivent pas se trouver là par exemple).

# PARTIE CLIENT



Premier écran sur lequel l’utilisateur arrive. Il doit rentrer son identifiant et son mot de passe pour se logguer puis cliquer sur le bouton Valider.

Figure . 1er écran



4 onglets sont disposés en bas de l’écran.

* L’écran principal contient la liste des mini-jeux. Si on clique sur un mini-jeu on arrive sur le jeu pour y jouer.
* Possibilité de se déconnecter
* Possibilité d’avoir accès à un historique des parties, des gains des pertes (à voir en fonction du temps)
* A propos : lien vers page FB, playstore pour noter l’application, copyright créé par Igor et Maxime UTT 2013/2014.

## Sécurité et risques d’attaques (valable pour l’ensemble de l’application) :

* Force brute : combinatoire, dictionnaire.  
  Solution : Bannir l’utilisateur après un certain nombre de tentatives (3 par exemple) avant qu’il puisse réessayer de se connecter. Obliger l’utilisateur à avoir un mot de passe robuste : minimum de caractères, avec majuscules, minuscules, caractères spéciaux.
* Social Engineering : Obliger l’utilisateur à avoir un mot de passe robuste : minimum de caractères, avec majuscules, minuscules, caractères spéciaux.
* Injection SQL : Vérifier les données entrantes : liste blanche, noire. Utiliser des requêtes paramétrables (pas de construction dynamique). Chiffrer/encoder les données entrantes. Donner les privilèges minimaux. Ne pas afficher les messages d’erreur.
* Authentification : attention au mot de passe comme dit plus haut, **stockage pas sécurisé** (chiffrer mdp dans BDD) Hashage + salage -> 1 sel par mot de passe comme ça il est plus long de trouver tous les mots de passe + captcha éventuellement…**, vol token**, man in the middle (-> tunnel SSH), cookie, vol de session
* **vol de session** 98% faille des appli mobiles (cause : transmission des données en clair, temps expiration de session, gestion des sessions non valides (logout), session id trop facile ; solution : connexion sécurisé SSL/TLS, infos données en plus à la session).
* **Tokens de validation :** ajout d’un jeton d’identification aléatoire que le client doit renvoyer à chaque requête. Vérification d’une page à l’autre.
* XSS (Cross site scripting) : injection de script côté client et l’utilisateur fait confiance au site et rentre ces identifiants.

Solution : Vérifier et filtrer les données. Désactiver scripts. Interdire l’accès aux cookies par Javascript.

* CSRF Cross Site Resource Forgery ou one-click attack : Utilisation de l’identité de l’user (cookie). Envoyer les requêtes http de la part de l’user. Pas besoin de l’autorisation de l’user.

Solution : Contrôle sur le domaine d’origine. Utilisation de tokens de validation.

* DDOS mais pas vraiment de solution à notre niveau et le risque est minime

# CONCLUSION SECURITE :

* Vérifier les données entrantes
* Désactiver les modules non utilisés
* Utiliser les librairies vérifiées et originales
* Protéger et chiffrer les données sensibles
* Utiliser SSL et les jetons d’accès

SERVEUR :

* Minimiser les données et les permissions et ne pas afficher les messages d’erreurs ni les versions ou en indiquer une fausse ! (serveur BDD etc). Cela permet d’éviter l’exploitation de failles connues. Les hackers devront utiliser des failles inconnues (notion de 0-day).

RESEAU :

* Utiliser Tunnel SSH pour qu’il ne soit pas détecté. Evite le Man in the middle.