CATÉGORIE TECHNIQUE

Parc des Marêts - Rue Peetermans, 80 - 4100 Seraing

Dans l'Optics d'un partage :

Application iOS de partage événementiel de photos.

Travail de fin d'études présenté en vue de l'obtention du grade de Bachelier en techniques graphiques finalité techniques infographiques

Année académique : 2015 - 2016

JÉRÉMY SMITH

TABLE DES MATIÈRES

AVANT PROPOS

« Tout ce qui suit résulte de ce qui est. »
-- Moi

J'ai longuement réfléchi quant au format de cet imprimé. Et ce ne sont pas les idées les moins originales qui ont manquées. Pourtant, de toutes celles qui m'ont traversées l'esprit, c'est la plus simple que j'ai retenue. Bien que la pensée d'un petit format, m'est pendant un temps, je dois bien l'avouer non négligeable, restée en tête j'ai préféré faire quelque chose dont j'avais envie. Et j'ai depuis toujours voulu tenir entre mes mains un livre que j'aurai moi-même réalisé. Autrement dit, vous avez là bien plus qu'un simple projet de fin d'étude mais un rêve à part entière.

Vous découvrirez au fil de ces pages mon cheminement, mes problèmes, mes solutions et parfois (voire souvent) mes contournements. Ce n'est ni plus ni moins que l'histoire de ma 3e année bien mouvementée…

Je dédierai quelques mots pour les remerciements en fin d'ouvrage, mais par respects pour les concernés, je me dois d'en remercier certains avant tout ! D'abord à Camille pour l'idée originelle du sujet. Ainsi que ma maman et mes amis / collègues et professeurs sans qui il ne fait aucun doute que ces mots n'existeraient pas. Merci à vous Adrien, Delphine, Arnaud et Pierre.

Enfin et surtout toutes les lignes du présent livre sont dédiées à Mémé, partie sans prévenir...

GENÈSE

« Au commencement il n'y avait rien »
-- Bible, Genèse I v1

PRÉMISSES D'UNE RÉFLEXION

La volonté de trouver un sujet de PFE ne remonte pas seulement au début de cette 3e année. Non, il faut remonter même avant que ne commence la première. Lors de la journée portes-ouvertes, où j'ai rencontré ceux qui allaient devenir mes futurs enseignants. C'est en fin de matinée que j'apprends un peu l'organisation des cours et des années. L'information essentielle étant celle d'un projet de fin d'étude à présenter devant un jury. Dès lors, la tâche de trouver une idée était ancrée quelque part dans mon esprit. Je ne prétends pas y avoir souvent songé, mais de temps en temps, cette pensée me passait par la tête. Cependant, à ce moment, je ne connaissais pas très bien la formation et je pensais que, dès lors qu'il était relativement conséquent, n'importe quel projet web ferait l'affaire. Ça tombait très bien, j'entreprenais tout juste de développer mon propre framework MVC SwithMVC[1].

Parfait, j'avais mon idée, ma motivation, et l'irrépressible envie d'être en 3e année pour me plonger intégralement

1 Pour plus d'informations, se rendre sur le site de la documentation : www.mvc.swith.fr

dans ce projet un peu fou. Mais évidemment, ce n'est pas si « simple ». Si ma mémoire ne me fait aucun défaut, je n'ai pas eu d'autres idées que celle-ci. Une personne, dont l'identité ne me revient pas (ah ! peut-être bien que ma mémoire me fait défaut finalement), m'avait rapidement parlé d'un concept de dictionnaire médical avec un fonctionnement tout aussi complexe que les termes qu'il tenterait de définir. Le projet était ambitieux certes, mais un « je-ne-sais-quoi » a fait que je l'ai vite écarté. Peut-être, ai-je là commis une belle erreur ?

Je n'ai pas été très productif concernant les idées de PFE. Et je ne sais plus à quel moment exactement, mais j'étais décidé à réaliser une application pour iPhone, pour plusieurs raisons. La première étant de découvrir un nouvel univers, sortir un peu du cadre du web où j'ai depuis longtemps trouver un véritable confort. Ce qui amène à la seconde raison : l'apprentissage d'un nouveau langage, le Swift[1]. Enfin, le nombre de possibilités étant infini, il y avait de quoi bien s'amuser avec un tel projet. Mais voilà, le problème étant que beaucoup de choses on été faites et qu'en plus de ça, une infinité de possibilités ne fait que rendre le choix plus difficile.

C'est alors qu'un soir, je demande à Camille si elle n'avait pas un concept qui « *valait 3 milliard* ». Rapidement elle a pensé à une application proche d'Instagram mais avec un fonctionnement original par l'utilisation d'un QR code. Bien que son idée n'était pas clairement formulée, j'ai tout de suite adhéré et beaucoup de choses me sont passées par la tête à ce moment là.

1 Langage de programmation créer par Apple pour remplacer l'objective C jusqu'à lors utilisé pour les applications iOS et OS X.

Rapidement, mes idées se sont précisées autour d'un partage de photos liées à un évènement. L'idée étant de pouvoir à la fois organiser et regrouper très facilement des photos venant de plusieurs personnes afin de ne résoudre au final qu'un seul problème.

ÉBAUCHE D'UNE RÉPONSE

Ne vous êtes-vous jamais retrouvé dans une situation similaire à celle-ci ? Vous allez à l'anniversaire d'un ami, passez une excellente soirée et prenez plusieurs photos. D'autres de vos amis en prennent aussi et vous aimeriez bien récupérer leurs photos. Pas beaucoup de solutions, on s'envoie chacun dans son coin les photos souhaitées. Ou alors, dans un soucis de simplicité, une personne récupère tout et s'occupe de mettre en ligne (fallait pas tirer la courte paille !) tout un dossier via le service de son choix et chacun viendrait prendre ce qui lui plaît. Je pense que nous sommes d'accord, ce n'est pas pratique ! Tel est le problème que j'ai tenté de résoudre. Et ce en inversant simplement le processus.

Le principe de mon application serait de permettre à une personne, principalement l'organisateur de l'événement ; dans mon exemple un anniversaire, de créer un dossiers dans lequel tout le monde pourra ajouter des photos. Ce dossier sera facilement partageable grâce à l'utilisation d'un QR code. Ainsi, ceux qui le souhaitent, pourront scanner ce code, récupérer le dossier et ajouter ses propres photos, ainsi que récupérer celles déjà présentes.

MA MOTIVATION

Ceux qui me connaissent le savent, je suis un passionné. J'ai depuis toujours été attiré par les arts visuels, le design et la technologie, l'informatique en particulier. Devinez alors ma joie quand j'ai découvert l'existence d'un métier qui liait le tout… L'infographie était une révélation ! Puis comme tout le monde j'ai rapidement eu des préférences ainsi que des facilités. Il m'est plus facile de coder un site que d'en réaliser le design et pourtant j'aime vraiment me poser toutes les questions qu'il faut pour concevoir une interface. J'aime avoir un processus de réflexion sur des problématiques auxquelles personne ne pense normalement mais qui, pour peu qu'elles soient bien prises en compte, peuvent grandement faciliter le quotidien.

Sans oublié mon goût pour les défis, je m'en suis trouvé un de taille. Apprendre un nouveau langage, sur une plateforme que je ne connais pas, du moins dans son fonctionnement technique car je l'utilise quotidiennement. C'est d'ailleurs dans cette *Optics* que j'ai voulu créer une application mobile, pour connaitre un peu mieux cet outil que j'utilise tous les jours : mon iPhone.

J'ajouterai par la même occasion une corde à mon arc car avoir une expérience dans l'application mobile peut ouvrir des portes dans le domaine du web et c'est une conséquence à prendre en compte.

Enfin j'étais aussi curieux de connaitre les différences de problématiques entre interfaces web et mobile. La réflexion allait-elle être la même ?

UN ÉTÉ CHARGÉ

« Il faut parfois courir avant d'apprendre à marcher ! »
-- Tony Stark

Je ne sais pas si ma méthode de départ a été la bonne. Mais aujourd'hui je ne la regrette pas car elle m'a permis d'avoir très rapidement un aperçu des difficultés que je pourrais rencontrer et d'évaluer les points qui me prendraient le plus de temps. J'avais l'idée principale de mon application dès la fin de la seconde année. Je disposais donc d'un peu d'avance pour expérimenter certaines choses sans pour autant craindre de foncer droit dans un mur. Telle à été ma méthode au départ :

PREMIÈRES EXPÉRIMENTATIONS DE SWIFT

Ma motivation étant à bloc, j'ai dès le début des vacances d'été commencé à mettre les mains dans le cambouis. Pas de temps à perdre pour apprendre ce nouveau langage qui, je dois bien le reconnaitre, me séduisait depuis un petit moment et ce pour plusieurs raisons. La première étant une facilité de prise en main avec une syntaxe épurée, verbeuse mais pas trop.

```
print( "hello World!" )
```

Pas de balises comme pour le php ou l'html. Pas de nécessité d'ajouter un ; à la fin de chaque instruction. De plus la syntaxe, dans sa globalité, se rapproche un peu de celle du CoffeeScript[1].

Comme pour tous les autres langages que j'ai appris par moi-même, j'ai commencé par suivre des tutoriels, pour me familiarisé avec les bases et l'environnement[2]. C'est alors sur les conseils de M. Delnatte que j'ai regardé les cours de l'université de Stanford, disponibles gratuitement sur iTunes[3]. Bien qu'ils soient basés sur une ancienne version du Swift, ils ont été une excellente base et ce malgré une certaine difficulté à tout comprendre (du fait que ce soit en anglais et que mon niveau est plus que moyen).

Cependant ce qui m'a le plus aidé au début, ce sont les très excellents tutoriels disponibles sur le site http://www.hackingwithswift.com qui m'ont vraiment apportés les premières bases de mon application et m'ont aidés à voir un peu plus clairement l'architecture d'une application iOS. Ainsi j'ai débuté à manipuler des données, jouer entre différentes vue, créer une petite interface.

ESSAIS D'APPELS REST

Derrière ce titre accrocheur mais néanmoins obscure pour les plus *néophytes* d'entre-vous se cache quelques chose de relativement simple. Mais nous ne sommes qu'au chapitre des expérimentations, je reviendrai sur les détails dans une

1 Le CoffeScript est langage qui permet d'écrire du JavaScript, de manière plus simple, pour en savoir plus : http://coffeescript.org.

2 Je préciserai les outils et technologies utilisés dans le chapitre dédié : Technologies.

3 Pour visualiser les vidéos : https://goo.gl/Ot52Qa.

prochaine partie. Pour le moment il vous suffit simplement savoir que mon application n'est pas autonome. J'entends par là, que seule, elle ne pourrait rien faire. Elle va devoir communiquer avec un serveur pour récupérer des données. Les appels REST sont justement cette communication.

Ainsi, il me fallait trouver comment avec Swift je pouvais communiquer avec un serveur disant. Par chance c'est tout à fait possible et même très simple. Il y a une fonction, très complète qui fait ça très bien : `dataTaskWithRequest()`. Pour faire mes tests, je n'avais pas de serveur distant. C'est alors l'API[1] de GitHub qui m'a servi de cobaye. Je me suis alors amusé à afficher mes informations GitHub. Vous pouvez aussi le faire ! Essayez dans un navigateur en tapant l'url suivante :

```
https://api.github.com/users/SwithFr
```

Vous devriez avoir un résultat proche de la capture d'écran ci-dessous.


```
{
    login: "SwithFr",
    id: 4257500,
    avatar_url: https://avatars.githubusercontent.com/u/4257500?v=3,
    gravatar_id: "",
    url: https://api.github.com/users/SwithFr,
    html_url: https://github.com/SwithFr,
    followers_url: https://api.github.com/users/SwithFr/followers,
    following_url: https://api.github.com/users/SwithFr/following{/other_user},
    gists_url: https://api.github.com/users/SwithFr/gists{/gist_id},
    starred_url: https://api.github.com/users/SwithFr/starred{/owner}{/repo},
    subscriptions_url: https://api.github.com/users/SwithFr/subscriptions,
    organizations_url: https://api.github.com/users/SwithFr/orgs,
    repos_url: https://api.github.com/users/SwithFr/repos,
```


Si tel est le cas, bravo ! Vous avez effectué un appel à l'API de GitHub tout comme moi durant mes tests. Bien entendu

1 Application Programming Interface. En (très) bref, un outil qui permet selon une URL de renvoyer des données.

c'est plus simple dans un navigateur qu'avec Swift. Mais vous pouvez comprendre le principe. On appelle une URL, on reçoit des données. D'ailleurs c'est quoi ces données ?

MANIPULATIONS JSON

Je vous le donne en mille… Suspense insoutenable… ces donnée sont du Json[1] ! C'est un format textuel de données issu des objets en JavaScript qui a la particularité d'être léger, facile à lire et écrire. Ce n'est ni plus ni moins qu'un ensemble de clés et de valeurs.

J'ai ici rencontré une première difficulté. Il faut savoir que le Swift est un langage dit typé. C'est-à-dire que les objets de ce langage sont associés à un type. Prenons un exemple simple avec le nombre `42`. Vous avez plusieurs manières d'interprété ce nombre. En tant qu'entier ou alors en tant que chaine de caractères représenté par les caractères `4` et `2`. L'un est un nombre l'autre est un « *mot* ». Alors évidement comme ça, ça ne change pas grand chose. Mais en programmation, il peut en ressortir des résultats inattendus. En voici un exemple absolument imaginaire mais qui je pense peut vous donner une idée du problème. Procédons à une simple addition : `42 + 2`. Si ce sont des nombres on obtient `44`. Mais si ce sont des « *mots* » ? Qu'obtient-on ? Le mot `422` ? Voilà donc le problème des types. En Swift il existe plein de types. Il y a les chaines de caractères, les entiers, les décimaux… Mais pas de type Json.

La manipulation de données au format json est toutes fois possible mais vraiment pas pratique ! Il faut écrire beaucoup

1 JavaScript Object Notation.

de lignes de code (environ une dizaine) pour convertir du json au format qui nous convient. Voyez par vous-même ce simple exemple[1] pour afficher l'âge d'une personne.

```
var json: Array!
do {
  json = try NSJSONSerialization.JSONObjectWithData(JSONData, options: NSJSONReadingOptions()) as? Array
} catch {
  print(error)
}

if let item = json[0] as? [String: AnyObject] {
  if let person = item["person"] as? [String: AnyObject] {
    if let age = person["age"] as? Int {
      print("Dani's age is \(age)")
    }
  }
}
```

Ici il faut imaginer que `JSONData` ait pour valeur quelque chose comme ça :

```
[
  {
    "person": {
      "name": "Dani",
      "age": "24"
    }
  },
  {
    "person": {
      "name": "ray",
      "age": "70"
    }
  }
]
```

Et c'est un exemple simple ! Imaginez un peu avec des données plus complexes ça serait invivable ! Par chance, d'autres ont réfléchi à une solution et il existe la librairie (fichier apportant des fonctionnalités supplémentaires) `SwiftyJSON`[2] qui rend la manipulation du json beaucoup plus simple !

1 Exemple tiré du site : www.raywenderlich.com/120442/swift-json-tutorial
2 https://github.com/SwiftyJSON/SwiftyJSON

MARK$_I$: PROTOTYPE

Si je résume, je sais à présent faire des appels à un serveur pour récupérer des données au format json, je sais à présent manipuler ces données pour les afficher et enfin les bases du Swift commencent à rentrer… Ne serait-ce pas à un feu vert pour passer à l'élaboration d'un premier prototype ?

Pour être honnête, j'étais assez confient à ce moment. Je n'avais pas vraiment eu de difficultés jusque là et j'aimais de plus en plus travailler avec cette nouvelle technologie. C'était un réel plaisir à tout moment de me demander « *comment fait-on ça en Swift ?* » J'étais en analogie constante avec ce que je connaissais des autres langages et on ne peut pas vraiment dire qu'ils soient fondamentalement différents, au contraire et c'est ce qui m'a aidé à avancer si rapidement au début.

Comment commencer un prototype fonctionnel sans API pour me renvoyer des données ? En effet, il me faudrait à terme, avoir une API et une base de donnée pour stocker mes informations, sur mes utilisateurs etc. Ce que je n'avais pas à ce stade. Pas de problème ! Ce n'est pas le plus important. Ce que je voulais vraiment c'était surtout avoir quelque chose qui me donnerait une idée de ce à quoi pourrait aboutir mon application, techniquement parlant. C'est pour quoi à ce moment, il n'y aucune réflexion sur le design ou sur l'interface. Donc j'ai fait semblant en intégrant des données fictives comme si je les avait reçues d'une API. Il ne me restait plus qu'à les manipuler, les afficher, les formater (par exemple pour les dates qui ne sont pas directement au format « *vendredi 13 mai* » mais plutôt `2016-05-13 11:57:07`. Et les transférer entre les vues. Puisque mon application avait pour

but de créer des évènements et d'y ajouter des photos, j'ai donc implémenter ces fonctionnalités. Pour ainsi dire j'avais en fait dans les 30% de mon application de fonctionnel et ce assez rapidement pour me mettre plus en confiance !

J'aurai aimé vous montrer des aperçus de ce que ça pouvait donner, mais dans un affrontement acharné avec une housse de couette récalcitrante, mon disque dur top secret avec toutes mes archives sécurisées (*aka* TimeMachine) est mort au combat, emportant avec lui tous les plans du Mark$_I$…

PREMIÈRES ANALYSES

Il ne faut pas croire que je n'ai eu aucune difficulté à réalisé ces ridicules petits 30%. Au contraire ! Certes, mon avance a été plus rapide que ce que j'avais pensé, mais elle n'a pas été facile pour autant. C'était un mal bien nécessaire, car si j'avais eu à passer cette étape durant l'année scolaire ça aurait été encore plus difficile.

Toutefois cette étape m'a permis de mettre en évidence une très grande lacune : l'anglais. Beaucoup des tutoriels, pour ne pas dire tous, que j'ai suivis étaient en anglais. Et certain aspects du langages Swift, pourtant simples, m'ont parus bien compliqués à comprendre du fait de mes difficultés avec cette langue.

D'autres difficultés me sont apparues. Plus techniques. L'architecture d'une application iOS, qui pourtant est une architecture MVC[1], m'est aujourd'hui encore un peu obscure.

1 Architecture que je maitrise assez bien puisque, je vous le rappelle, j'ai développé un framework MVC en php.

Car elle suit une logique que je n'ai pas tout à fait saisi et qui m'a par la suite posé l'un des deux plus gros problèmes que j'ai du surmonter.

Après ça je savais un peu plus à quoi m'attendre. Je savais que ça ne serait pas si simple, que j'aurai des choix à faire, que je devrais priorisé certaines fonctionnalités. Mais mes idées se précisaient au fur et à mesure que mes connaissances augmentaient. J'avais à présent une idée bien concrète de ce que ferait l'application, il ne restait plus qu'à formaliser tout ça.

CAHIER DES CHARGES

« Que l'on me donne six heures pour couper un arbre, j'en passerai quatre à préparer ma hache. »

-- Abraham Lincoln

Assez foncé tête baissée ! Maintenant mon sujet est validé, il est grand temps de passer aux choses sérieuses et d'être organisé. Nous avons bien assez tôt été avertis que le temps manquerait et que l'organisation était un luxe dont on ne pouvait se passer. Rapidement il nous a été demandé d'établir un cahier des charges détaillé des fonctionnalités de notre projet. De fait il serait amené a évoluer et le mien a pas mal bougé. Le voici dans sa version (presque) finale.

FONCTIONNALITÉS PRINCIPALES

Avant toutes choses, je dois préciser que, pour le moment du moins, j'ai du laisser de côté, l'idée d'utiliser un QR code sous les conseils d M. Delnatte. Car c'est semble-t-il assez compliqué à mettre en place et que c'est le point qui l'inquiétait le plus après avoir lu mon premier *brief* . Je n'ai pas abandonné l'idée pour autant. Elle reste une fonctionnalité que je m'efforcerai à mettre en place par la suite.

En raison des problèmes rencontrés durant cette année (vraiment difficile…) j'ai perdu pas mal de temps pour me recentrer et il en résulte que certaines fonctions, signalées par un * dans la liste suivante, on été mises de côté (temporairement) pour me focaliser sur les objectifs primordiaux au bon fonctionnement de l'application.

- Créer un compte utilisateur pour se connecter à l'application ;
- Permettre la modification des informations du profil ;
- Créer un dossier pour un évènement ;
- Partager cet évènement via les outils de partage de l'iPhone ;
- Ajouter / télécharger des photos à un évènement ;
- Rejoindre un évènement ;
- Supprimer les photos que l'on a partagé ;
- Commenter les photos ;
- Définir une liste d'amis ;
- Retirer un utilisateur de la liste d'amis ;
- Restreindre un évènement à certaines personnes * ;
- Signaler une photo comme inappropriée *.

FONCTIONNALITÉS FUTURES

En plus des deux fonctions marquées par un * dans la liste précédente, d'autres viennent s'ajouter pour une éventuelle

évolution future. Mais comme certaines nécessitent un compte développeur payant elles ont été écartées des objectifs principaux.

- L'organisateur d'un évènement peut refuser des participants ou leur empêcher d'ajouter des photos ;
- Mettre en avant des photos envoyées par un utilisateur dans la liste d'ami ;
- Avertir de la mise à jour d'un évènement par des notifications push (nouvelle photo, nouveau commentaire ou participant, etc.) ;
- Application web ;
- Compatibilité Androïde.

PUBLIC CIBLE

! !! ! ! !! Pas vraiment de public défini. Ce qui ne facilite pas la tâche

INTERFACE

À présent que les fonctionnalités sont définies, il faut commencer à prévoir l'interface. Et pour cela rien de mieux que des *wireframes* pour avoir une idée de comment seront agencés les différents éléments et se représenter le design de l'application.

Si vous avez visité mon portfolio[1], vous avez certainement remarqué la simplicité du design épuré. Car s'il est un prin-

1 www.swith.fr

cipes dicté par Dierter Rams[1] auquel j'adhère totalement c'est bien celui-ci : « *good design is as little design as possible* ». Autrement dit, un bon design est le design le plus minimum possible. Pas besoin d'un nombre incalculable d'effets en tout genres. De surcouches d'éléments, que ce soit pour combler le vide ou non ou encore apporter du détail. L'important c'est que l'information, le message que l'on veut faire passer, soit compris de la manière la plus simple et efficace possible. Je ne prétends pas avoir tout compris avec mon site mais je pense avoir trouvé le style graphique qui me convient le mieux : simplicité et clarté.

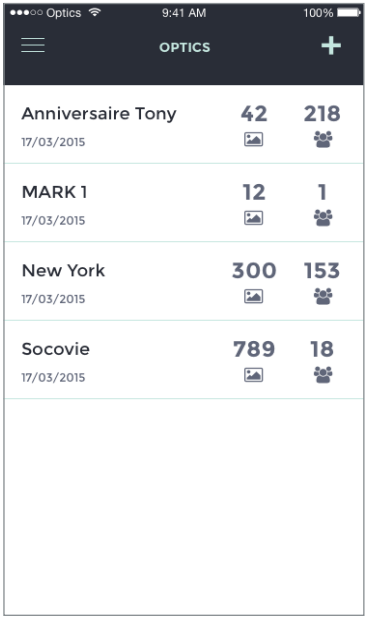
J'ai de la chance ! c'est aussi ce que préconise Apple dans ces *guidelines*[2] : « *providing clarity is another way to ensure that content is paramount in your app* ». Il suffit de regarder les changements opérés depuis iOS 7 pour comprendre. Il n'est pas nécessaire d'en faire trop. Il faut rester clair dans notre design. C'est ce que j'ai tenté de respecter tout au long de ma réflexion sur l'interface de mon application.

Voici à gauche la première version wireframe de la vue listant les évènements auxquels participe un utilisateur.

1 Dieter Rams, est un designer industriel allemand contemporain né le 20 mai 1932.

2 Ensemble de règles et principes rédigés par Apple dans le but d'aider à la conception d'interface sur iPhone. Pour en savoir plus : https://goo.gl/71LwsC.

On remarque très vite qu'il y a un problème d'équilibrage typographique qui vient perturber l'interface et apporter du poids sur la partie droite. J'ai rapidement corrigé ce problème en réalisant le Mark$_{II}$.



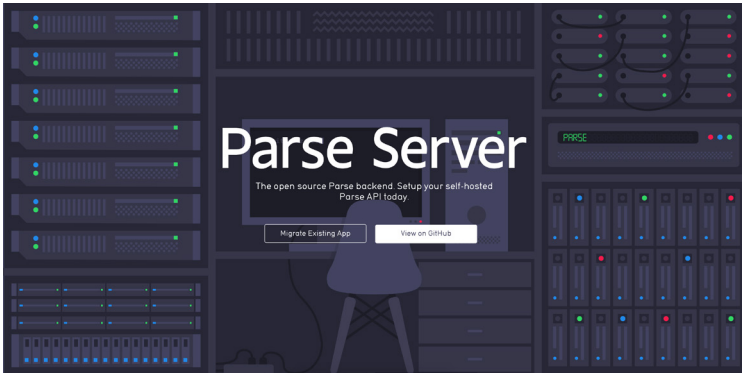
Et voici à droite la version Mark$_{II}$ de la même vue. C'est tout de suite mieux, non ? En ce qui concerne la simplicité, c'est le nombres d'informations disponibles. Seulement l'essentiel. Le nom ainsi que la date de création de l'événement de l'événement, le nombre de photos partagées et de participants. Sur un tel listing les informations supplémentaires seraient superflues.

Quelques mots sur la navigation. Bien que le menu ne comporte que trois items, ce qui serait facilement intégrable dans une *tab bar,* j'ai préféré faire appel au célèbre icône hamburger. Facilement identifiable comme accesseur à un menu. Pour l'icône d'ajout en haut à droite. Il parle de lui même. On est sur un listing d'événement, que pourrait-on ajouter d'autre qui rendrait cet élément ambiguë ? Cette vue est importante car c'est la première que l'utilisateur voit après s'être connecté. C'est elle qui va guider l'utilisateur sur l'utilisation de l'application. Si l'on sait utiliser cette vue, on saura utiliser toute l'application car elle suivra le même fonctionnement sur toutes les autres vues.

Je ne détaillerai pas chaque vue de l'application. Mais pour les plus curieux, vous trouverez en annexe, les wireframes complets du Mark II.

MARK II : COMMUNICATION ÉTABLIE

Dans la section précédente, vous avez pu apercevoir une vue du Mark II. Mais la nouveauté ici ne se limite au simple remplacement des wireframes par quelques couleurs et éléments graphiques ! Souvenez-vous du premier prototype à seulement 30% fonctionnel mais qui ne communiquait pas avec un serveur. Il était grand temps d'aller plus loin. Le problème est que je n'avais toujours pas d'API, il me fallait donc un outil qui en mettrait une à ma disposition. Ça tombe bien, c'est exactement que ce propose Parse[1].



C'est un outil dédié aux applications mobiles. Le service est assez simple et propose aux développeurs une gestion de base de données ainsi que tout un tas de librairies facilitant l'accès à celles-ci et la manipulation des informations. Tout comme SwiftyJSON mais avec beaucoup plus de fonctionnalités.

1 https://parse.com

Plus fort encore, Parse mettait à disposition des outils tels qu'un système d'enregistrement / connexion d'utilisateurs. Ainsi en quelques lignes de code seulement, vous pouviez intégrer une authentification sans trop vous fracturer le crâne !

Ça fonctionnait plutôt bien, j'avais pu reprendre mon premier prototype et supprimer toutes les données fictives qui me servaient pour tester et les remplacer par les méthodes que donnait Parse pour récupérer les données depuis l'API de Parse. Et assez rapidement j'étais passé des 30% au double.

Cependant j'ai rapidement abandonné cet outil. Mais pourquoi diable ai-je décidé une telle tournure pour mon projet ? Vous vous en douter surement, plusieurs raisons se cachent derrière ma décision.

Premièrement, il faut reparler des types en programmation. Parse, aussi utile soit-il encapsule tous les données qu'il peut nous renvoyer dans des objets (comprenez types) qui lui sont propres. C'est très malin puisque ça permet entre autre de greffer des fonctionnalités supplémentaires sur chacun des objets. Autrement dit, ajouter des manipulations ou actions possibles et différentes sur les différents types. Le problème est que ces objets sont des éléments fournis par et pour Parse. Par exemple, une image de Parse est du type `PFFile`. Mais en Swift c'est du type `UIImage`. Vous le voyez venir le problème ? Il faut convertir les objets Parse en objets exploitables en Swift. Et au final c'est un comportement qui ne me correspondait pas.

Deuxièmement, comme je l'ai dit, j'aime les défis. Et les trucs tout faits, clés en main, généralement, je ne suis pas

fan. Il me démangeait de créer ma propre API sur laquelle j'aurai un total contrôle. Et qui, pour des raisons de simplicité, ne renverrait qu'une seule chose : du json. En plus à cette période de l'année nous avions en cours de RIA, commencé à développer une petite API en nodejs[1]. Et je dois bien l'avouer, moi qui jusqu'à présent ne jurais que par le PHP, j'ai été littéralement séduit par cette technologie très simple à mettre en place. De plus le cours nous avait fourni des bases amplement suffisantes pour se lancer dans un tel développement. Ce qui ajouterait un petit plus à l'ensemble de mon projet qui certes ne casse pas trois briquets à une mouette (comment c'est pas ça l'expression ?). Mais au moins, j'aurai appris un nouveau langage et développé ma propre API, sans me contenter d'outils tout faits. Et quelques soit l'issue de mon PFE, je resterai fier de ce que j'aurai produit.

SECONDE ANALYSE

À ce stade mon application commençait vraiment à prendre forme. Mais elle avait un énorme problème. Le code que j'avais écrit était exécrablement mal organisé et presque aussi bien structuré que la tour de Pise. Beaucoup de répétitions, aucune optimisation. Ce n'était pas vraiment volontaire, mais mon code évoluait en fonction de mon apprentissage des méthodes et du langage. Le but premier étant de mettre en pratique ce que j'apprenais et de réutiliser quand j'en avais besoin des petites parties de code. La prochaine étape serait de structurer tout ça correctement.

Il faut aussi savoir que je rencontrais aussi de gros problèmes sur la mise en place de l'API et l'installation du serveur qui

1 Technologie permettant d'utiliser du JavaScript côté serveur et ne plus être limité au côté client. Se référer au chapitre Technologies.

l'accueillerait. Aussi, comme je l'ai dit plut tôt, l'architecture même d'une application iOS me mettait face au problème le plus étrange que j'ai eu au cours du développement. Mais j'y reviendrai dans un chapitre spécifique aux difficultés rencontrées.

OPTICS

« Que la lumière soit ! Et la lumière fut. »
-- Bible, Genèse I v3

La lumière a ceci de particulier que, contrairement à ce que tout le monde croit, sa vitesse n'est pas constante mais invariante. Elle n'est pas constante puisqu'elle diffère selon le milieu où elle se propage. Par exemple, elle va un peu moins vite dans l'eau que dans l'air. Plus intéressant encore, sa vitesse, limite absolue, est indépendante de la source qui l'émet et de l'observateur ! Ce qui fait appelle à une certaine ouverture d'esprit (rien a voir avec une fracture du crâne, paraît-il) qui n'est pas la plus facile, j'en conviens. Mais imaginez un instant que vous puissiez courir à une vitesse très proche de celle de la lumière, soyons fous, disons 99% de sa vitesse. Il est tentant de penser que si vous faites la course avec un rayon lumineux, il vous suffirait d'accélérer un peu pour gagner le pour-cent manquant et ainsi rattraper ce faisceaux. Mais non ! Puisque la vitesse de la lumière est indépendante de son observateur, quelque soit votre vitesse, le rayon lumineux se déplacera, par rapport à vous, à la vitesse de la lumière. C'est difficile à concevoir, je sais et pourtant ! Fascinant n'est-ce pas ? Si le sujet vous intéresse, je vous recommande les nombreuses conférences d'Étienne Klein, directeur du CEA (commissariat à l'énergie atomique), disponibles sur YouTube. Personnellement, je suis fasciné par la physique quantique et comment elle a su aussi rapidement

chambouler notre perception de l'univers. Sans la lumière, ingrédient *sine qua non* à l'obtention d'une photo mon application n'aurait pas lieu d'être. Quoi de plus normale alors de lui donner pour nom, celui (anglophone) de la branche physique qui traite de cet objet insaisissable : Optics ?

THE BLACK HOLE

Il est des moments dans la vie auxquels nous aimerions nous soustraire mais ils nous happent et nous retiennent à la manière des trous noirs d'où rien ne s'échappe, pas même la lumière. Sans le vouloir, ni même le savoir, je me suis approché de ce *Gargantua* et fatalement j'ai succombé à sa force gravitationnelle, sombrant au début de cette année 2016, dans les ténèbres de ses entrailles. Dramatiques conséquences, je ne me suis pas présenté aux examens de janvier et ai perdu toute motivation à fournir le moindre effort pour quelque raison que ce soit.

Pour ne rien n'arranger, j'avais passé un nombre incalculable d'heures sur les deux problèmes majeurs rencontrés jusque là (voir le chapitre : Houston we have a problem!) et à chaque fois que je travaillais sur ce projet, c'était pour faire face à un

problème. Ce n'était tout simplement plus possible. Et une pause, voire une hibernation, était fortement envisageable et conseillée. Pour réfléchir et me sortir la tête de cet environnement où ne résonnaient que bugs et problèmes.

C'est ce que j'ai évidemment fait. Souffler un peu, beaucoup même pour revenir, non pas à neuf, j'aurai bien aimé mais revenir avec assez de motivation pour terminer ce projet. Il m'aura fallut du temps et cela se paie au prix de fonctionnalités que j'ai du écarter, pour rattraper le temps perdu. C'est pourquoi les *features* secondaires prévues dans le cahier des charges ne seront pas intégrée. Je ferai de mon mieux pour tout de même avoir une plateforme web et/ou un site de présentation qui tiennent la route dans les délais qui me restent, promis j'essaierai.

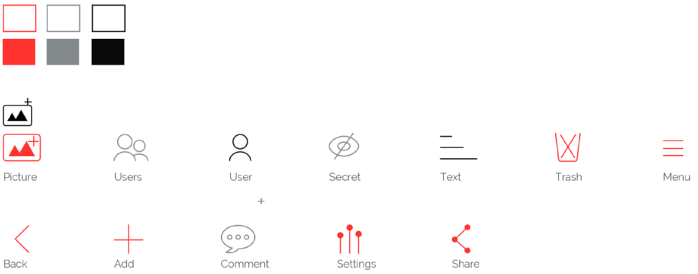
Assez déprimé, c'est parti pour un nouveau départ !

MARK$_{III}$: NOUVEAU LOOK

Par nouveau look, je ne veux pas seulement parlé du design, qui au final n'a pas grandement changé, seulement la palette de couleurs qui a évoluée. Mais comme l'a révélé ma dernière analyse. L'architecture même de mon projet était bancale et non professionnelle. Je suis donc reparti de zéro ! J'ai commencé par analyser mon code, repérer les répétitions pour cibler ce qui pourrait être *refactoré*. Mes fichiers étaient mal séparés tout se mélangeait. Je ne suivais aucune convention de nommage et de syntaxe, etc. Bref il fallait tout reprendre et c'est ce que j'ai fait pendant les 3 mois de stage. Tous les soirs je me replongeais un sur Optics. Et la motivation est revenue peu à peu.

Revenons en à la palette de couleurs. Pour ce nouveau départ, je voulais changer de thème, moins coloré et un peu plus sombre, peut-être en raison de l'expérience que j'étais en train de vivre, je ne sais pas. Voilà le résultat :

Moins de couleurs pour plus de simplicité, telle est la règle que je me suis fixée. Par la même occasion, j'ai pris la liberté de m'écarter des guidelines d'Apple. En n'utilisant pas nécessairement les icones fournis d'office. Mais en créant mes propres éléments d'interface utilisateur. Afin de personnifier



un peu plus l'application qui jusqu'à présent ne se distinguaient autrement que par le logo.

TROISIÈME ANALYSE

TECHNOLOGIES

Parler des choix techniques. Environnement iOS, nodejs, docker. Dire que vais essayer d'être le plus clair et imagé possible.

ENVIRONNEMENT DE DÉVELOPPEMENT

Parler d'Xcode sa prise en main, sont interactivité, le simulateur.

SWIFT

Parler du langage et detailler certains points, pourquoi mon model ? Mes conventions ?

NODEJS

Dire ce qui m'a plu dans le langage (j'étais enfermé dans l'interactivté que permet js coté client et arriavais pas a voir comment utilisé coté serveur)

DOCKER

Choix technologique (beaucoup entendu parlé donc voulu tester).

Encore un défi car réputé pour être dur à prendre en main mais une fois maitrisé, ça devient vraiment cool.

GIT FLOW

adoption de git flow. Systeme de convention et j'aime les conventions ! aide à la structuration de notre workflow. ça force à faire les choses petit à petit et part pas dans tous les sens.

SWIPES

gestionnaire de tache simple mais agréable à utilisé. Pas 3000 fonctions, juste ce qu'il faut. En plus app mobile et bureau, synchro gratuites.

HOUSTON WE HAVE A PROBLEM!