
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	5
GENÈSE	7
PRÉMISSES D'UNE RÉFLEXION	7
ÉBAUCHE D'UNE RÉPONSE	9
MA MOTIVATION	10
UN ÉTÉ CHARGÉ	11
PREMIÈRES EXPÉRIMENTATIONS DE SWIFT	11
ESSAIS D'APPELS REST	12
MANIPULATIONS JSON	14
MARK _I : PROTOTYPE	16
PREMIÈRES ANALYSES	17
CAHIER DES CHARGES	19
FONCTIONNALITÉS PRINCIPALES	19
FONCTIONNALITÉS FUTURES	20
PUBLIC CIBLE	21
INTERFACE	21
MARK _{II} : COMMUNICATION ÉTABLIE	24
SECONDE ANALYSE	26
OPTICS	28
THE BLACK HOLE	29

MARK_{III} : NOUVEAU LOOK.....	30
Les couleurs.....	31
Les icones.....	33
Navigation.....	34
Logo.....	36
Dilemme des boîtes modales.....	37
Inspirations.....	38
TECHNOLOGIES & DÉVELOPPEMENTS.....	41
ENVIRONNEMENT DE DÉVELOPPEMENT.....	41
Top model.....	45
Conventions.....	46
NODEJS.....	46
EXPRESSJS.....	47
GIT FLOW.....	48
SWIPES.....	49
HOUSTON WE HAVE A PROBLEM!.....	49
CYCLE MONSTRUEUX.....	49
DOCKER DONNE MAL AU CŒUR.....	49

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 3^e année bien mouvementée...

Je dédierai quelques mots pour les remerciements en fin d'ouvrage, mais par respect 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

PRÉMISSSES D'UNE RÉFLEXION

La volonté de trouver un sujet de PFE ne remonte pas seulement au début de cette 3^e 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 SwithMVC¹.

Parfait, j'avais mon idée, ma motivation, et l'irrépressible envie d'être en 3^e 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 Swift¹. 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 ont 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 milliards* ». 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 créé par Apple pour remplacer l'objectif 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 événement. 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 souci 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 dossier 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 connaître 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 connaître 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 a 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 reconnaître, 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 CoffeeScript¹.

Comme pour tous les autres langages que j'ai appris par moi-même, j'ai commencé par suivre des tutoriels, pour me familiariser avec les bases et l'environnement². C'est alors sur les conseils de M. Delnatte que j'ai regardé les cours de l'université de Stanford, disponibles gratuitement sur iTunes³. 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 vues, créer une petite interface.

ESSAIS D'APPELS REST

Derrière ce titre accrocheur mais néanmoins obscur 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 CoffeeScript 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 distant. 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'API¹ 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 enten-

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

du 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 que sont ces données ?

MANIPULATIONS JSON

Je vous le donne en mille... Suspense insoutenable... ces données 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'interpréter ce nombre. En tant qu'entier ou alors en tant que chaîne de caractères représenté par les caractères 4 et 2. L'un est un nombre l'autre est un « *mot* ». Alors évidemment 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 chaînes 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 exemple¹ 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) `SwiftJSON`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 confiant 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.

Mais 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 avais 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 implémenté 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 réaliser 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. Mais 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 certains aspects du langage 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 MVC¹, m'est aujourd'hui encore un peu obscure.

1 Architecture que je maîtrise 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 saisie et qui m'a par la suite posée 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 prioriser certaines fonctionnalités. Mais mes idées se précisaient au fur et à mesure que mes connaissances augmentaient. J'avais à présent une vision 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 à 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 de 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*. Mais 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, ont 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 événement ;
- Partager cet événement via les outils de partage de l'iPhone ;
- Ajouter / télécharger des photos à un événement ;
- Rejoindre un événement ;
- 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 événement 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 portfolio¹, vous avez certainement remarqué la simplicité du design épuré. Car s'il est un des


¹ www.swith.fr

principes dicté par Dieter Rams¹ 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.



Anniversaire Tony 17/03/2015	42	218	
MARK I 17/03/2015	12	1	
New York 17/03/2015	300	153	
Socovie 17/03/2015	789	18	

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, le nombre de photos partagées et de participants. Sur un tel listing les informations supplémentaires seraient superflues.

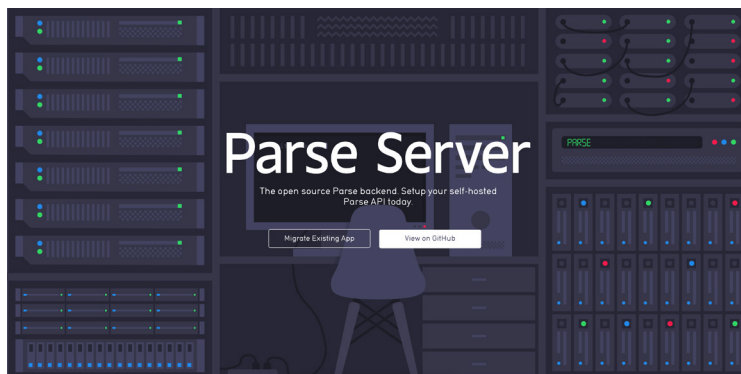
OPTICS		
Anniversaire Tony 17/03/2015	42 	218
MARK 1 17/03/2015	12 	1
New York 17/03/2015	300 	153
Socovie 17/03/2015	789 	18

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 ce que propose Parse¹.



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 bibliothèques facilitant l'accès à celles-ci et la manipulation des informations. Tout comme SwiftJSON 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 doutez sûrement, 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 nodejs¹. 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

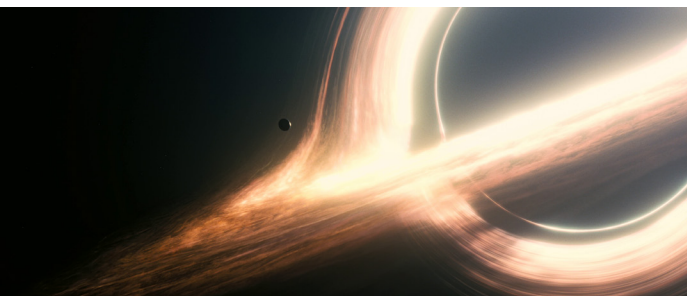
« Que la lumière soit ! Et la lumière fut. »

-- 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 faisceau. 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 normal 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.

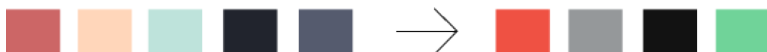
C'est ce que j'ai évidemment fait. Souffler un peu, beaucoup même pour revenir, non pas à neuf, j'aurai bien aimé mais revenir avec assez de motivation pour terminer ce projet. Il m'aura fallu du temps et cela se paie au prix de fonctionnalités que j'ai dû écarter, pour rattraper le temps perdu. C'est pourquoi les *features* secondaires prévues dans le cahier des charges ne seront pas intégrées et que j'ai fait l'impasse sur d'autres. 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 parler du design, qui au final n'a pas grandement changé, seulement la palette de couleurs qui a évolué. Mais comme l'a révélé ma dernière analyse, l'architecture même de mon projet était bancal 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 peu sur Optics et la motivation est revenue peu à peu. Mais nous y reviendrons dans le chapitre sur les technologies.

Les couleurs

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. Mais voilà le résultat :



Moins de couleurs pour plus de simplicité, telle est la règle que je me suis fixée. C'est après avoir longuement *spammé* la barre d'espace sur le site www.coolors.co, un générateur de palettes vraiment très pratique, que je suis arrivé sur un échantillon se rapprochant de ce que je voulais. Je l'ai alors un peu modifié pour avoir le résultat ci-dessus.

Je me suis efforcé de n'employer les différentes couleurs que dans des cas précis. Ainsi le rouge orangé, se réfère le plus souvent aux actions (boutons, items de navigation, etc.) sinon pour marquer un contraste (bordure sur les listings par exemple). Le gris quant à lui n'est utilisé que pour du texte dynamique ou certains icones. Le texte blanc étant réservé aux éléments statiques tels que les labels. Enfin le noir, ne sert que pour le fond, la barre de navigation étant l'exemple le plus marquant.

Typographie

Le choix de la typographie a été relativement difficile ! Je suis passé par Helevtica puis Montserrat, Avenir Next pour finir avec Raleway¹. Mes premiers choix n'étaient pas réflé-

¹ Caractère initialement dessiné par Matt McInerney dans une seule grasse

chis, c'était le résultat par défaut. Par la suite j'ai cherché de l'inspiration, pour guider ma réflexion c'est alors que j'ai installé l'application Medium¹, ressource d'articles tous aussi excellents les uns que les autres et dont le site fait déjà preuve d'un usage typographique remarquable et l'application ne fait pas exception. C'est à mon sens l'une des applications au contenu riche qui a su le mieux gérer l'équilibre typographique. Installez cette application ! Et voyez par vous-même. Pour les titres FF Kievit² ainsi que ITC Charter³ pour le contenu. Seulement l'application montre beaucoup de contenus et de différents types, d'où la nécessité d'un jeu serif/sans-serif.

En ce qui me concerne, je n'ai pas tant de diversités, j'ai donc rapidement convenu qu'une seule fonte serait amplement suffisante. J'avais d'abord retenu Avenir Next, redessinée par le regretté Adrian Frutiger décédé en septembre de l'année dernière et Akira Kobayashi. Étant parfaitement adaptée à la lecture sur écran elle était un bon point de départ.

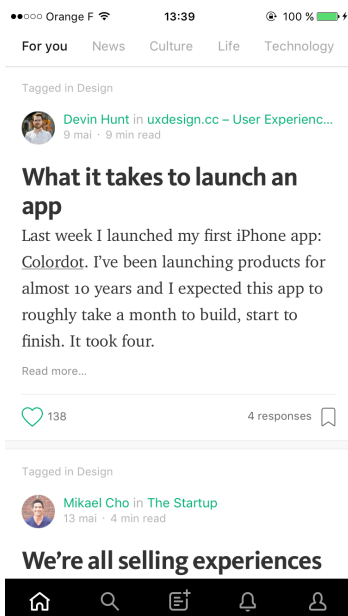
Mais après une analyse plus approfondie, j'ai remarqué des petits détails disgracieux qui ne me plaisaient que moyennement. Premièrement Avenir Next a une force de corps légèrement plus faible ce qui lui donne un effet écrasé, surtout en petite taille.

légère puis étendue en 2012 par plusieurs contributeurs.

1 <https://medium.com>.

2 <https://www.fontfont.com/fonts/kievit>.

3 <https://www.myfonts.com/fonts/itc/charter/>.



Optics Optics

Raleway 24pt

Avenir Next 24pt

Certaines combinaisons de lettres telles que « *rt* » ne sont pas très élégantes. Ajouté à cela des terminaisons trop courtes ou trop brusque, ma préférence c'est portée sur Raleway.

rt j rt j

Raleway

Avenir Next

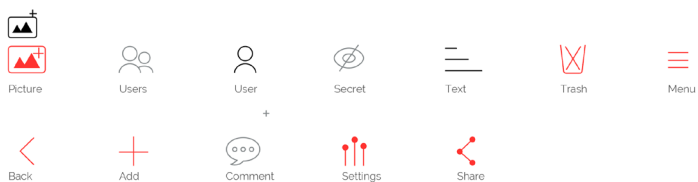
Dernier point typographique sur lequel j'aimerais parler plus en détail : les différentes tailles. Apple recommande¹ de ne jamais aller en dessous de 11pt. Dans un soucis de simplicité je n'utilise que principalement deux tailles différentes : 16 et 20pt. Car une fois encore, je n'ai pas beaucoup de contenu distinctif. Ces deux valeurs suffisent à contraster les différents contenus. Pour les choisir, je me suis fortement inspiré des conseils que l'on peut trouver sur le site Designcode².

Les icônes

En retravaillant le design, j'ai pris la liberté de m'écarter des guidelines d'Apple en n'utilisant pas nécessairement les icônes fournis d'office. Mais en créant mes propres éléments d'interface utilisateur. Afin de personnifier un peu plus l'application.

1 En référence au guidelines : <https://goo.gl/TAk7Lc>.

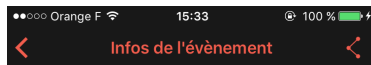
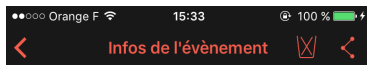
2 <https://designcode.io/iosdesign-typography>.



Pour être honnête, je n'ai pas cherché d'inspiration pour ces éléments. Je voulais en effet quelques chose de vraiment personnel sans influence. Bien entendu c'est impossible, on reconnaîtra vite l'icône représentant un utilisateur ou celui des commentaires. Ce que je voulais surtout c'était des traits simples et fins.

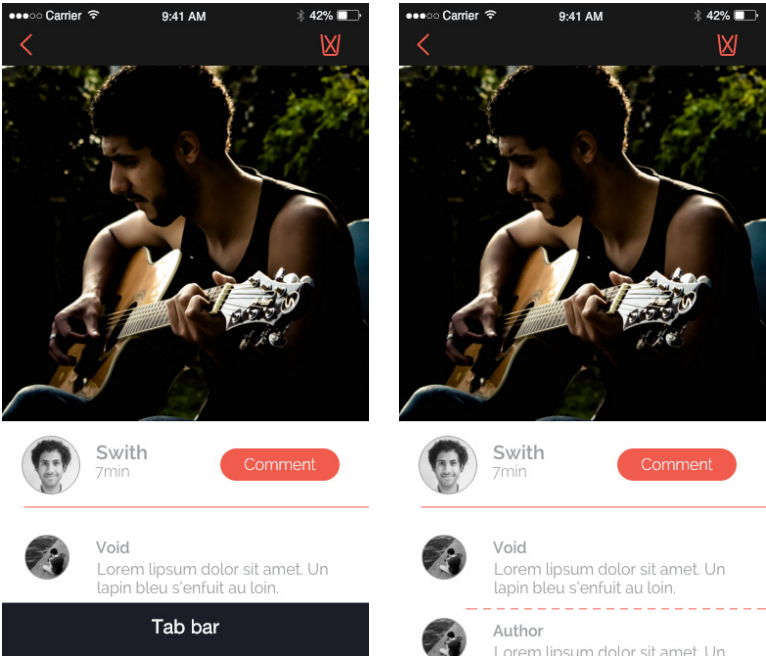
Navigation

Nous avons déjà rapidement abordé le sujet de la navigation mais je vais maintenant aller un peu plus loin. J'ai cherché à la soulager au maximum, pour ne pas être encombré d'éléments secondaires. Le but étant de ne faire apparaître que les interactions importantes et les rendre accessibles immédiatement. La navigation est aussi contextuelle, c'est-à-dire qu'elle diffère selon si on est propriétaire de la ressource ou non. Par exemple, si je n'ai pas créé un évènement, je ne pourrai pas le supprimer, donc l'icône de suppression n'apparaît pas.



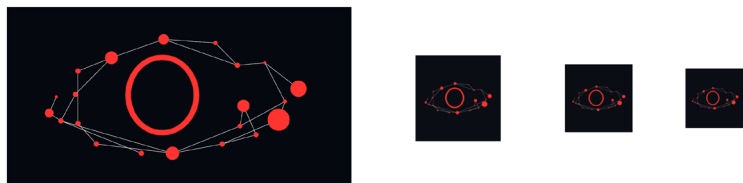
Comme nous le verrons par la suite, la navigation se trouve en haut de l'écran mais sur iOS, elle peut tout aussi bien être au bas de l'écran dans ce qu'on appelle une *tab bar*. À ce pro-

pos il y a un article¹ très intéressant sur Medium, préconisant une navigation avec la *tab bar* car plus facile d'accès sur mobile et énonce aussi quelques règles à suivre pour faciliter le parcours au sein d'une application. Cependant j'ai choisi de ne pas suivre cette recommandation car bien que tout a fait justifiée, elle se prête beaucoup plus à une navigation par onglet (d'où le nom, *tab* en anglais signifie onglet). Ce qui ne se prête pas forcément à mon projet. De plus, le but étant de voir des photos, je ne voulais pas perdre, en quelques sortes, de l'espace ce qui pourrait éventuellement entraîner un comportement inattendu. Ainsi on pourrait ne pas se douter que l'on peut scroller pour voir plus de contenus.



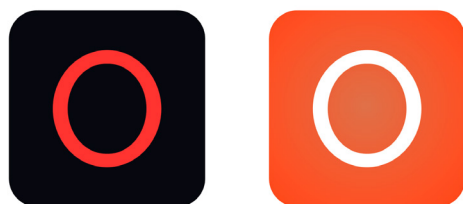
1 <https://goo.gl/KGFJ1O>.

Logo



Il est grand temps de parler un peu de cet œil aux points reliés telle une constellation sur son ciel noir. L'astronomie me passionnant tout autant que la physique des particules, je ne pourrais nier une certaine influence. Cependant ce n'est pas le point le plus fort que j'ai voulu partager avec ce logo. La notion que j'ai surtout souhaité mettre en avant est celle de partage et de connexion entre les utilisateurs.

Il n'est pas parfait, une fois réduit ça devient illisible, les traits sont trop fins et le contraste s'estompe. J'ai fait plusieurs tentatives pour avoir quelque chose de plus adaptable sans être satisfait des résultats. Les plus acceptables que j'ai réalisés sont les suivants :



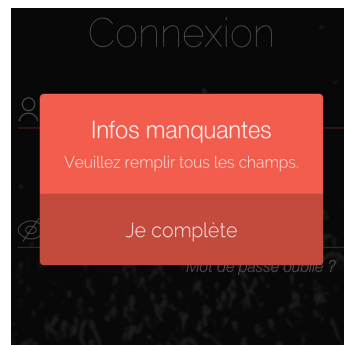
Mais pour le coup, leur simplicité ne me convenait vraiment pas. Je me suis peut-être (trop) attaché au premier ce qui ne le rend pas définitif pour autant. Il n'est pas à l'abri d'une évolution, tout comme l'application dans son ensemble.

Dilemme des boites modales

À la manière de popups, il est possible de faire apparaître une petite alerte lors d'une action importante telle qu'une suppression ou encore une confirmation de choix.



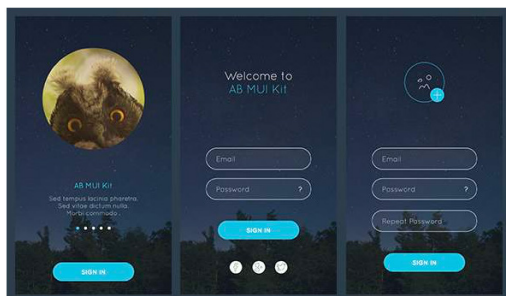
Cette interface est celle proposée par défaut, très facile à mettre en place ce qui en fait la plus largement répandue et celle que l'on s'attend à voir dans de telles situations. Et c'est là que se pose le dilemme car l'envie de personnifier ces interactions est très forte mais prend du temps et risque de perturber les utilisateurs. Il faut dire aussi qu'elle ne s'intègre pas toujours très bien au design de l'application et on préférerait une boîte modale plus proche de l'exemple ci-contre. Le dilemme se pose toujours car à l'heure actuelle, je n'ai toujours pas arrêté mon choix. Mon hésitation se balance entre la facilité d'intégration et l'esthétisme.



Génie du design, je n'ai jamais eu besoin d'une quelconque inspiration ! Comment ça ce n'est pas vrai ? Si, si je vous assure ! Bon d'accord j'avoue Dribbble¹ et Pinterest² on été mes meilleurs amis pendant un bon moment. Puis plus je regardais des designs, moins j'arrivais à me décider. Quelque chose me plaisait puis l'instant d'après j'avais changé d'avis. et ça ne s'améliorait pas avec le temps. Honnêtement, il y a tellement de belles réalisations que les regarder ne fait que nous sentir de plus en plus pathétique.

J'ai donc du faire un choix et ne me limiter qu'à de simples petits éléments plutôt qu'à des vues complètes que j'aurais éventuellement adaptées à mon goût (ce qui reviendrait à dégrader l'originale dont je me serais inspiré).

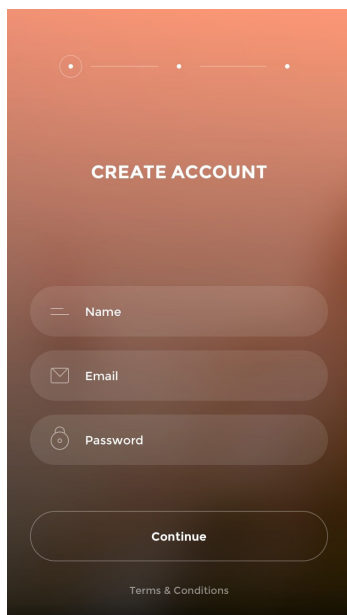
D'abord l'idée de reprendre un peu partout l'image de fond ma été soufflée par cette image prise d'un PSD gratuit. J'ai trouvé intéressant le fait de rappeler simplement par le fond l'ambiance générale de l'application.



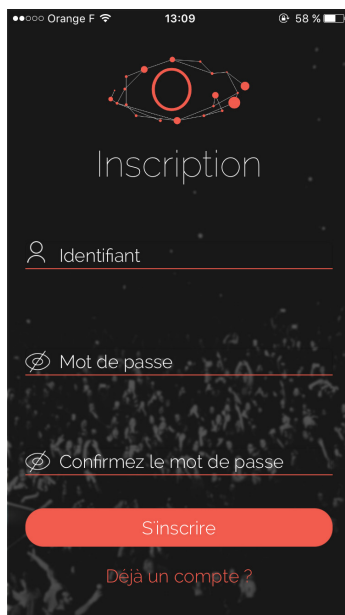
<http://goo.gl/MJgFSz>

- 1 <https://dribbble.com>.
- 2 <https://www.pinterest.com>.

C'est sur un autre kit gratuit que je me suis basé pour intégrer des icônes dans les labels pour la connexion et la création d'un compte. C'est un petit détail qui vient servir de rappel une fois que l'utilisateur a commencé à taper dans le champ texte.



<https://goo.gl/8fdzQw>



Bien entendu, je me suis aussi très fortement inspiré de l'application Instagram puis qu'elle est ce qui se rapproche le plus d'Optics, surtout pour le listing des photos et si le temps me le permet, je m'inspirerai également de la plateforme web d'Instagram pour le site d'Optics.

TROISIÈME ANALYSE

J'arrive enfin à quelque chose de plus définitif et plus personnel. Toutes les fonctionnalités que je souhaitais vraiment intégrer sont opérationnelles et dans l'ensemble, il ne me reste que quelques détails à régler. Par exemple trouver une icône pour le bouton de téléchargement des photos qui puisse être affichée par dessus les images et ce quelques soit les couleurs et contrastes de celle-ci.

Je suis content d'avoir repris le projet depuis le début. J'ai ainsi réussi à résoudre certains problèmes en étant plus structuré dans ma manière de développer et surtout en ayant pris le temps de réfléchir à ce que je voulais vraiment. Cette pause m'aura coûtée du temps et des sacrifices pour les fonctionnalités, mais je pense qu'elle a été entièrement bénéfique.

TECHNOLOGIES & DÉVELOPPEMENTS

« Sans technique, un don n'est rien qu'une sale manie. »

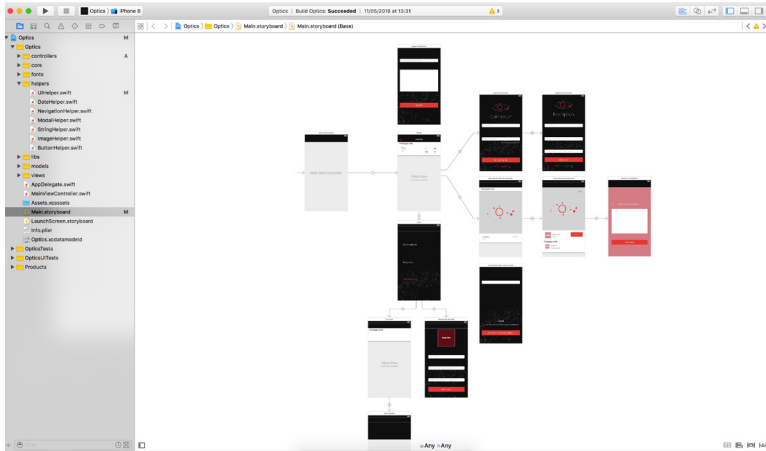
-- Georges Brassens

Dans cette partie j'aborderai mes choix techniques. Pourquoi ai-je choisi d'utiliser telle technologie plutôt qu'une autre ? Je parlerai également du développement en lui-même en abordant des points plus précis, tels que les conventions que je me suis fixées, de la manière dont j'ai architecturé mon application. Je tâcherai, le plus possible, de rester clair et d'imaginer au maximum mes propos pour ne pas vous perdre avec un jargon de développeur. Puisque toute chose n'est rien sans son contexte, j'explicitai également mon environnement de développement qui a son importance. Enfin, très rapidement je m'attarderai sur des logiciels qui m'ont servi tout au long de l'élaboration de cette application.

ENVIRONNEMENT DE DÉVELOPPEMENT

Commençons par ce qui n'a pas véritablement été un choix mais une obligation : Xcode. Pour développer une application iOS malheureusement nous n'avons pas le choix. Il faut utiliser l'éditeur d'Apple pour compiler (transformer les lignes de code en quelque chose que l'on peut exécuter)

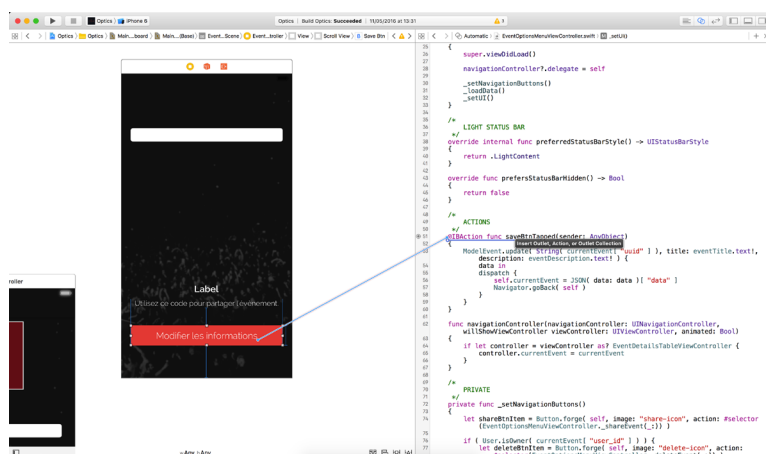
l'application. Mais il faut savoir que c'est un peu plus qu'un simple éditeur. Il met à disposition plusieurs outils très pratiques pour aider au développement comme l'*assistant editor* dont je parlerai juste après. Car il faut d'abord que je vous parle du storyboard.



C'est certainement de loin l'outil le plus agréable à utiliser. Il permet en quelques clics de disposer les vues de votre application, de leur ajouter des éléments (boutons, labels, inputs, tableaux...) et de les lier entre elles par simple glisser-déposer. Ainsi vous pouvez cliquer sur un bouton et le lier à une vue pour que lorsque l'on appuie sur ce bouton il affiche la vue adéquate. C'est presque magique ! Nous pouvons alors rapidement visualiser l'architecture de notre application et voir dans leur ensemble les parcours possible pour aller d'un endroit à un autre.

Le storyboard s'utilise souvent de paire avec l'*assistant editor* qui permet de mettre côte à côte le storyboard avec la page

de code qui lui correspond. Alors dit comme ça, ça peut vous sembler inutile mais détrompez-vous, c'est extrêmement pratique. Le storyboard permet une stylisation simple des éléments. Pour un bouton par exemple, on peut définir, la taille, la couleur de fond, la typographie, etc. mais ça reste malheureusement limité, on ne peut pas définir de bords arrondis (Oui je suis d'accord, ça craint !). Mais il ne faut pas croire que ce n'est pas possible pour autant, il faut juste le faire avec des lignes de code (comme on le ferait en CSS). Et bien avec l'*assistant editor*, on peut drag & drop notre bouton sur la page de code qui correspond à la vue à laquelle il appartient, pour créer une variable qui fera référence à ce bouton. Et si vous vous souvenez quand je vous parlais des types en Swift, il existe un type pour les boutons : `UIButton` qui renferme plusieurs méthodes pour modifier le bouton, dont une pour définir des bords arrondis. Mais on peut aussi aller plus loin et définir des actions spécifiques plutôt qu'une variable. Ainsi on peut procéder de la même manière pour créer une fonction qui se déclenchera lorsqu'on appuiera sur le bouton.



Xcode dans son ensemble fait preuve de beaucoup d'avantages, il est très pratique et son interactivité est agréable à utiliser notamment avec l'*assistant editor* et le storyboard. De plus il a une gestion des erreurs qui fait gagner beaucoup de temps car il propose même de résoudre certains problèmes tout seul (erreur de syntaxe, oublie de paramètres, etc.).

Cependant il n'est pas parfait non plus. En tant qu'éditeur de code il n'est pas optimal, surtout quand on a l'habitude du très complet PhpStorm. Il est malheureusement difficilement personnalisable et la gestion des raccourcis clavier n'est vraiment pas pratique.

Je terminerai par un rapide mot sur le simulateur, car Xcode va également de paire avec ce dernier. On peut alors lancer notre application et tester sur différents appareils et leurs versions (iPhone 4, 4s, 6, iPad 2, etc.). C'est très pratique d'autant plus que c'est vraiment rapide. Pas besoin de brancher son téléphone, d'attendre que l'application soit installée et perdre du temps à chaque fois.

SWIFT

Une fois encore, ce n'est pas vraiment un choix, quoi que j'aurai bien pu faire de l'Objective-C mais puisque Apple a créé son propre langage pourquoi ne pas l'utiliser ? C'est le choix le plus judicieux à mon sens puisqu'il est toujours préférable d'utiliser un langage natif (langage spécifique à la plateforme avec laquelle on va l'utiliser). Mais l'Objective-C n'est pas perdu pour autant et pourrait faire l'objet d'un futur défi (notamment avec la compatibilité Androïde). Cependant comme je l'ai dit plus tôt, la syntaxe du Swift me faisait

un peu de l'œil depuis quelques temps contrairement à celle de l'Objective-C qui semble carrément barbare !

```
// Swift
let image = UIImage( named: "test.jpg" )

// Objective-C
UIImage* image1 = [UIImage imageNamed:@"MyImage"];
```

Exemple pour créer une constante contenant l'image test.jpg.

Honnêtement il ne m'aura pas fallu beaucoup de temps pour décider du Swift et je pense que ça peut facilement se comprendre avec l'exemple ci-dessus.

Top model

Rappelons que nous sommes dans une architecture MVC (pour *Model View Controller*). Alors très rapidement, il s'agit d'une manière de structurer son code pour faciliter le développement. Le *model* sert à récupérer des données, le controller à les traiter, et la *view* à afficher le tout. Bon, c'est une définition très sommaire mais amplement suffisante pour comprendre la suite. Comme vous devez vous en douter, ici je vais m'attarder sur la partie *model*. Imaginez que c'est un petit robot esclave sous-payé¹ auquel vous donnez des ordres tels que : « *va chercher X!* » où *X* représente des données sur une API.

À partir de maintenant deux possibilités s'offrent à vous. Soit vous apprenez à utiliser le robot d'un autre en lisant la notice soit vous construisez votre propre robot. C'est cette deuxième option que j'ai suivie. Non pas que je sois plus malin que les

¹ Toute ressemblance avec des robots existants ou ayant existé est purement fortuite.

autres mais c'est surtout que je n'aime pas me contenter de savoir utiliser quelque chose mais le plus souvent je préfère comprendre comment ça marche pour la simple et bonne raison, d'être un maximum autonome. Prenez l'exemple d'une voiture, beaucoup savent l'utiliser, conduire n'est pas si compliqué mais combien savent comment ça fonctionne vraiment ? Une telle connaissance vous fera économiser des allers-retours chez le garagiste... C'est dans cette Optics que j'ai préféré construire mon propre robot.

Conventions

NODEJS

Avant tout nodejs¹ qu'est-ce que c'est ? C'est une technologie relativement jeune qui permet de démarrer des serveurs JavaScript pour des applications en temps réel (messagerie instantanée par exemple). Ce qui veut dire qu'on va pouvoir utiliser du JavaScript à la place d'un langage plus habituel tel que le php pour générer des pages web. Cette technologie est intéressante de part son principe de fonctionnement basé sur des événements. En effet on va pouvoir (comme en JavaScript « classique ») définir des actions au déclenchement d'événements particuliers.

Si j'ai choisi d'utiliser nodejs plutôt que du php c'est tout simplement parce que ça a été une très agréable surprise. En effet j'étais cantonné dans mon langage de prédilection, le php et j'avais une vision du JavaScript limité aux interactions avec le client. Je ne voyais vraiment pas comment on pouvait utiliser un tel langage côté serveur. Puis voilà qu'en

¹ <https://nodejs.org/en/>.

cours de RIA, nous voyons comment développer une petite API basée sur nodejs et je suis littéralement séduit. C'est une approche tout à fait différente qui m'a vraiment donné envie d'en faire un peu plus. C'est là que j'ai eu l'idée de créer ma propre API puisque ça semblait si simple et ça l'est en effet. Je n'ai vraiment eu aucune difficulté à créer mon API. Cependant nodejs n'est pas un langage, il nous faut tout de même écrire du JavaScript et pour ça j'ai utilisé le framework Expressjs¹.

EXPRESSJS

C'est un framework pour nodejs qui offrent des fonctionnalités qui permettent d'aller un peu plus vite dans le développement. L'une d'elles nous intéresse en particulier, la possibilité de générer simplement des routes (URLs). Si pour vous cette dernière phrase a autant de signification que si je l'avais écrite en chinois, voici une petite explication.

Souvenez-vous qu'on récupère des données depuis une API en appelant une URL spécifique. Les routes dont je parle, sont justement les URLs que l'on va appeler. Cependant elles n'existent pas par magie, il faut bien les créer et faire en sorte que chacune d'entre elles nous renvoie une information précise. C'est là qu'intervient Expressjs, en nous permettant de structurer notre API, et d'effectuer telle ou telle action en fonction de la route que l'on demande.

Peut-être vous demandez-vous pourquoi ai-je utilisé un framework existant plutôt que d'en créer un moi-même puisque, comme je l'ai dit, je préfère comprendre comment

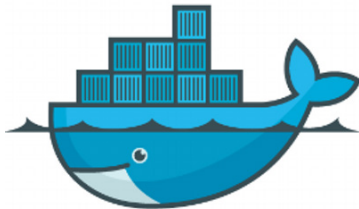
¹ <http://expressjs.com>.

ça fonctionne que de me contenter de savoir utiliser. C'est une excellente question ! Premièrement, c'est d'un tout autre niveau qui requiert des compétences que je n'ai pas (encore). Un framework, quelque soit son langage, ne se fait pas en cinq minutes (sauf si on s'appelle Adrien Leloup et qu'on est capable de faire la base d'un CMS¹ en un peu moins d'une journée). À titre d'exemple, pour SwithMVC, mon framework php, j'ai travaillé dessus pendant presque trois ans donc faire un framework pour nodejs, bien que l'idée me soit passée par la tête, n'était pas envisageable dans un délais aussi court. Il faut parfois savoir déléguer.

J'aurai aimé faire une critique plus approfondie de cette technologie mais ce serait forcément erroné car je n'ai pas de comparatif. Nous avons vu en cours une bonne base depuis laquelle je suis parti pour mon API sans vraiment chercher s'il n'y avait pas mieux ailleurs.

DOCKER

En y réfléchissant bien je pense que M. Delnatte est un Jedi maîtrisant la force. Pas d'autres explications que celle-ci car voilà encore une technologie que l'on a découvert au cours de RIA, dont j'avais certes entendu parlé mais qui me semblait tout aussi obscure qu'attrayante.



Docker² est une technologie dont le principe est basé sur un système de conteneurs. Il faut imaginer une boîte dans laquelle on va mettre tout ce

¹ <http://www.kabas.io>.

² <https://www.docker.com>.

dont on a besoin pour que notre application fonctionne. Car aujourd'hui une application a besoin de plus en plus de dépendances (comprenez logiciels tiers) pour fonctionner. Pour bien comprendre, imaginez que vous voulez cuisiner un bon plat. Vous avez besoin d'ingrédients donc vous allez les acheter et parfois il arrive que vous ne trouviez pas tout au même endroit, il vous faut vous rendre dans plusieurs magasins pour avoir tout ce dont vous avez besoin, pas pratique n'est-ce pas ? Vous ne seriez donc pas contre une boîte avec tous les ingrédients dedans et vous seriez prêt à suivre la recette en quelques minutes seulement. Eh bien c'est ça le principe de conteneur. Sans rentrer dans les détails, avec un fichier de configuration, nous allons pouvoir définir tous les ingrédients dont nous avons besoin pour pouvoir les utiliser par la suite.

Le problème de cette technologie c'est qu'elle est relativement difficile à prendre en main. Même si son fonctionnement paraît simple avec l'exemple que je viens de vous donner, il faut savoir que c'est beaucoup plus complexe que ça et qu'aujourd'hui encore, je ne serai pas capable de vous décrire vraiment en profondeur son fonctionnement. La preuve en est que même en cours nous avons dû abandonner Docker qui nous a fait perdre un temps considérable tant la mise en place était compliquée. Mais vous commencez à me connaître, j'ai vu là à défi de plus à surmonter et ça m'a donné encore plus envie d'utiliser cette technologie. Après coup j'ai sévèrement regretté ce choix car j'ai un problème qui m'a fait perdre vraiment beaucoup de temps !

GIT FLOW

À présent ce n'est pas une technologie mais plus une méthode de travail que j'ai adoptée dont je vais vous parler. En effet au cours de mon stage chez Novius¹, j'ai été amené à utiliser le système de version Git² avec GitHub³. Rapidement, le contrôle de version est un système permettant de sauvegarder l'état d'un fichier et d'en réaliser un historique. Ainsi on peut facilement revenir à une version précise de notre travail. On peut créer diverses branches qu'il faut voir comme des mondes parallèles dans lesquels on peut faire ce que l'on veut sans altérer les autres. C'est extrêmement pratique dans le cas d'un développement. Imaginez que vous ayez un code fonctionnel mais que vous voulez tester quelque chose qui implique beaucoup de modifications dans plusieurs fichiers tant que si au final votre test n'est pas concluant, il vous serait impossible de revenir assez loin en arrière pour retrouver votre code qui fonctionnait très bien. C'est un cas typique d'utilisation des branches.

Alors git flow qu'est-ce que c'est ? C'est en fait une sorte de convention qui va nous aider à structurer notre travail et faciliter notre utilisation de Git. Pour résumer simplement, c'est surtout une convention de nommage pour les branches pour s'y retrouver plus facilement. Sans aller plus loin dans les détails, car je ne pense pas que ce soit pertinent, si j'ai décidé de parler ici de cet outil c'est pour bonne raison. C'est au cours de mon stage que j'ai découvert l'utilisation de git flow et ça m'a fait prendre conscience d'une chose, c'est qu'en plus de faciliter l'utilisation de git, ça m'a permis d'être beaucoup

1 <http://www.novius.com>.

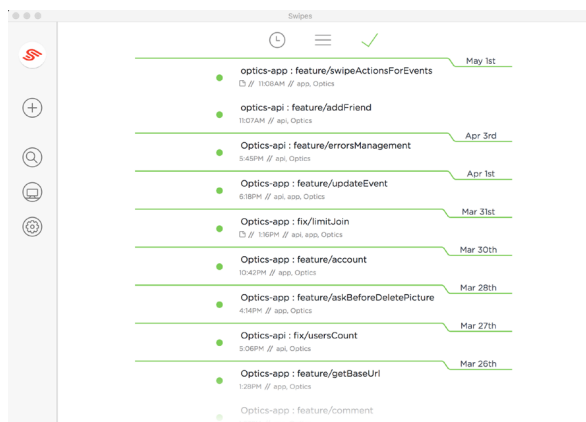
2 <https://git-scm.com>.

3 <https://github.com>.

plus organisé dans ma manière de coder. En effet, j'avais du mal à rester concentré sur une tâche en particulier. Par exemple, si j'étais occupé à développer une fonctionnalité et que je remarquais un problème à un autre endroit, souvent je m'arrêtais pour le corriger, puis ce faisant je remarquais souvent un autre problème, ou une meilleure façon de faire pour autre chose, etc. Puis quand finalement je revenais sur ce quoi de travaillais au départ, j'avais un peu perdu le fil. Git flow a changé tout ça. Le fait de créer des branches spécifiques, correctement nommées pour chaque chose m'a en effet aidé à être beaucoup plus rigoureux. Je restais plus facilement concentré sur une tâche et si je voyais un problème ailleurs, j'ajoutais une note à mon gestionnaire de tâches.

SWIPES

En parlant de gestionnaire de tâches, je vous présente Swipes¹. C'est outil simple mais agréable à utilisé. Il n'y a pas une infinité de fonctions, le strict minimum avec un design clair et intuitif.



1 <http://swipesapp.com/>.

HOUSTON WE HAVE A PROBLEM!

introduction des 2 pb rencontrés

CYCLE MONSTRUEUX

problème lors de la connexion, instanciation des vue qu'ils ne devrait pas.

DOCKER DONNE MAL AU CŒUR

