

公立はこだて未来大学 2015 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2015 System Information Science Practice
Group Report

プロジェクト名

フィールドから創る地域・社会のためのスウィフトなアプリ開発

Project Name

“Swift” Application Development Based on Field Research

グループ名

観光系グループ

Group Name

Tourism Group

プロジェクト番号/**Project No.**

3-A

プロジェクトリーダー/**Project Leader**

1013220 新保遙平 Yohei Shinpo

グループリーダー/**Group Leader**

1013068 岩見建汰 Kenta Iwami

グループメンバ/**Group Member**

1013001 池田俊輝 Toshiki Ikeda

1013068 岩見建汰 Kenta Iwami

1013167 山川拓也 Takuya Yamakawa

1013224 細川椋太 Ryota Hosokawa

1013228 横山翔栄 Shohei Yokoyama

指導教員

伊藤恵 奥野拓 原田泰 木塚あゆみ 南部美砂子

Advisor

Kei Itou Taku Okuno Yasushi Harada Ayumi Kizuka Misako Nambu

提出日

2016 年 1 月 20 日

Date of Submission

January 20, 2016

概要

本プロジェクトでは、フィールドを実際に調査してそこから問題点を見つける。そこで見つかった問題点を解決するために iOS アプリケーション（以降、アプリとする）を開発して、それにより地域・社会に貢献することを目標として活動を行っている。開発手法はアジャイル開発手法を用いる。素早くアプリを開発し、それに対するレビューを受けてより質の高いアプリを開発していく。

我々観光系グループは 2016 年春の北海道新幹線開業に伴い、観光産業に力を入れている木古内町をフィールドに設定した。5 月初旬に実際に木古内町に現地調査へ行き、どのような問題点があるのか、どのような魅力があるのかを調査した。フィールドワークから、観光情報が SNS や Web などに分散して分かりにくいという問題点があることに気づいた。また、観光客はスマートフォンで撮影した写真で思い出を振り返ることが少ないことに気づいた。そこから我々が話し合っ固めたアプリ案をティーチングアシスタント（以降、TA とする）や担当教員、企業講師の方々にレビューを受けながら、PDCA サイクルを用いて前期と後期で 4 回の開発を行った。前期の開発では、分散している観光情報の集約と、観光中に撮影した写真を自動的にアルバムにして思い出話を盛り上げるための機能「フォトストーリー」を提案した。しかし、7 月の中間発表会で、これはただ観光情報を表示しているだけでユーザが「行きたい」と思えないという課題と、写真を撮るモチベーションが高くないという課題が見つかった。後期では、前期の課題を解決するために、木古内で「できること」に着目した情報の表示、そして自分の撮った写真からオリジナルのリーフレットを作成する機能を提案し、実装を行った。この時、開発アプリの名前を「キーコ紀行」とした。12 月の成果発表会ではキーコ紀行のデモを行い、レビューを受けた。

成果発表会でのキーコ紀行の評価が高かったため、今後もメンバ間で開発を進めて 2016 年 2 月にはリリースすることを予定している。まずは木古内の町長や観光協会の人々にアプリをレビューしてもらうことが必要である。さらに、まだアプリ内に残っているバグ修正を行っていく。

キーワード 木古内町, 観光, アジャイル開発, iOS アプリ, 現地調査, レビュー, PDCA サイクル

(※文責: 山川拓也)

Abstract

In this project, at first, we investigate on the field and find problems from field survey. We develop iOS application (app) to solve the problems found from field survey. Then we have action with the goal of contributing to an area. We use Agile development process which is a software development technique. We do swift app development, and develop higher quality app by being reviewed for it.

We tourism group set Kikonai town as the field. The town began to lay emphasis on tourism industry due to opening Hokkaido Shinkansen in the spring of 2016. In the beginning of May, we surveyed what kind of problems and charms Kikonai has by going there in fact. From this, we found that the sightseeing information disperses in SNS and Web. Furthermore, tourists don't look back the photos which they took. Then, we did development four times totally with PDCA cycle in the first semester and the second semester while being advised of our app's ideas to Teaching Assistant (TA), teachers, and the professional instructors from companies. In the first semester, we proposed two functions: intensiveness of dispersing information and "photo story," making an album automatically from photos which users took during sightseeing in order to enliven recollections talk. However, in the midterm presentation of July, we found two problems that app only showed information about Kikonai and it couldn't make user raise the motivation to take photos. In the second semester, we proposed two functions: sightseeing information focused on "things to do in Kikonai" and making an original leaflet by using taken photos. Then we implemented them. In this time, we named this app "Ki-ko Kiko." In the final presentation in December, we gave a demonstration of it and have been reviewed from many people.

We continue to develop it and intend to release this app by February in 2016 because its evaluation was high opinion in the final presentation. At first it is necessary to have the mayor of Kikonai and tourism association review it. In addition, we will fix bugs in it.

Keyword Kikonai Town, Tourism, Agile Development Process, iOS Application, Field Survey, Reviews, PDCA Cycle

(※文責: 山川拓也)

目次

第 1 章	背景	1
1.1	観光産業と新幹線	1
1.2	木古内町と観光	1
1.3	課題の概要	1
第 2 章	到達目標	3
2.1	本プロジェクトにおける目標	3
第 3 章	プロジェクトの進め方	4
3.1	進め方の概要	4
3.2	開発の進め方	4
3.2.1	計画 (Plan) フェーズ	4
3.2.2	実行 (Do) フェーズ	6
3.2.3	評価 (Check) フェーズ	6
3.2.4	改善 (Act) フェーズ	7
第 4 章	開発準備	8
4.1	開発に使用したツール	8
4.2	勉強会	8
第 5 章	開発プロセス	9
5.1	第 1 サイクル (5 月 1 日～7 月 3 日)	9
5.2	第 2 サイクル (7 月 3 日～7 月 10 日)	11
5.3	第 3 サイクル (9 月 25 日～10 月 21 日)	14
5.4	第 4 サイクル (10 月 22 日～12 月 11 日)	17
第 6 章	キーコ紀行について	19
6.1	キーコ紀行の概要	19
6.2	「観光する」機能	19
6.2.1	カードリスト画面	19
6.2.2	マップ・詳細画面	20
6.2.3	写真撮影画面	21
6.2.4	テキスト編集画面	22
6.3	「振り返る」機能	23
6.4	「印刷する」機能	24
6.5	リーフレットについて	27
第 7 章	使用技術	30
7.1	データベース	30
7.2	カードリスト	31

7.3	マップ・詳細画面	33
7.4	カメラ	36
7.5	振り返る機能	37
7.6	印刷	40
第 8 章	今後の課題と展望	43
8.1	リリースについて	43
8.2	ステークホルダーへの報告会について	43
第 9 章	学び	45
9.1	グループ間の情報共有	45
9.2	計画管理の必要性	45
9.3	バージョン管理	46
9.4	ポスター	47
第 10 章	到達目標に対する評価	48
10.1	機能の評価	48
10.2	活動内容の評価	49
第 11 章	まとめ	50
11.1	まとめ	50
付録 A	その他新規習得技術	51
A.1	WebAPI	51
付録 B	活用した講義	52
付録 C	その他成果物	55
C.1	Git マニュアル	55
	参考文献	57

第 1 章 背景

1.1 観光産業と新幹線

2015 年 3 月に東京-金沢間を走る北陸新幹線が開業し、北陸の観光産業が活気づいている。それは交通の利便性が向上したことによって、石川県などの北陸の観光が各種メディアで取り上げられて注目されたことが要因に挙げられる。実際に新幹線開業後のゴールデンウィークには石川県金沢城公園に前年同期比 2.5 倍もの観光客が訪れたという例もあるほどで、観光産業における新幹線開業の効果は大きい。このように新幹線の開業は、その事業の規模から地域に注目を集め、交通の利便性が向上することによって観光客の増加を促す効果がある。

(※文責: 細川椋太)

1.2 木古内町と観光

木古内町は北海道道南地域にある山と海に囲まれた、漁業と酪農及び畜産を基幹産業とする人口 5000 人程の自治体である。この木古内町には、2016 年春に開業する北海道新幹線の停車駅ができる。それによって観光客の増加が見込まれることから、観光産業にも力を入れている。例えば、木古内町観光の基盤づくりとしては「観光交流センター・みそぎの郷きこない」の建設や駅前道路の整備が行われている。また、観光地としての知名度向上活動に関しては、木古内町マスコットキャラクターのキーコが木古内駅新幹線観光駅長としてイベントで PR を行うなどの活動が行われている。このように、観光産業を盛り上げる活動を積極的に行っている木古内町だが、観光情報は各メディアで分散して発信されており、観光客にとって必要な情報を見つけやすい状態であるとは言いがたい。現在観光情報は、木古内町観光協会の Web サイト [1] や Facebook、パンフレットなど様々なメディアで発信されている。様々なメディアが情報発信を行っているため、観光客は各メディアから多様性のある情報を得ることができる反面、情報の一覧性が低くなってしまっている。また、観光者向けパンフレットはあるものの、我々が 2015 年 5 月に 1 度目の木古内町フィールドワークを行った際には、木古内駅構内など観光客が簡単に見つけることのできる場所に置かれていなかった。そのため、その場で観光情報を得ることが難しくなっていた。これらの要因から、木古内町の観光の魅力が観光客や木古内町を知りたい人に伝わりづらいという問題がある。

(※文責: 細川椋太)

1.3 課題の概要

我々はアプリ開発を行うにあたり、大きく分けて 2 つの課題の解決に取り組む。まず 1 つ目は、観光情報のアクセス性改善と魅力の発信である。先述の、観光情報が各種メディアに分散しているため観光情報の一覧性が低くなってしまっている点を解決し、その情報を効果的に伝えることによって木古内町観光を魅力的に伝える。2 つ目の解決する課題は、観光した際に撮影した写真を振り返る機会が少ないことである。観光中にスマートフォンで撮影した写真を整理をしたり振り返る

“Swift” Application Development Based on Field Research

という機会があまりないという観光客は少なくないだろう。しかしこの課題を解決し写真を振り返る機会を増やすことができた場合、木古内町を思い返し再度観光へ行くリピーターの増加や、写真を通した思い出話から広がる口コミによって木古内町の認知度が高まることに繋がると考えられる。本プロジェクトでは上記2点の課題へ、観光アプリを開発することによってアプローチし、木古内町観光の満足度向上を図る。

(※文責: 細川椋太)

第 2 章 到達目標

2.1 本プロジェクトにおける目標

本プロジェクトは、木古内町へ来訪する観光客の満足度を高めることを目標としている。前述の通り、木古内町の観光情報は分散しており、そのアクセス性は高くない。そこで本プロジェクトでは、観光情報へのアクセス性を高めることによって観光客の負担を減らすほか、観光で得られる体験に付加価値を与えることで、木古内観光の満足度向上を実現する。アプリケーション設計に際しては、木古内町へのフィールドワークで得られた経験を重視し、メンバの実体験に基づくアプリケーション設計を行うことで、より木古内の現状に即したアプリケーションを開発することを目標とした。開発段階では本学講義の一つであるソフトウェア設計論で学習したアジャイル開発手法のうちスクラムを実践することで、敏速なアプリケーション開発を目指した。コーディングにあたっては言語に Swift を用い、これの習熟に努めるほか、Git や IDE、データベースなどの活用を通じて、アプリケーション開発のための技術を習得することを目標とした。これらに加え、後期では UI 設計や UX の検討など、機能面だけでなくユーザの体験、アプリケーションの使いやすさの面も考慮し、リリースを視野に入れた実用に耐えうるアプリケーションの開発を目標とした。

(※文責: 横山翔栄)

第3章 プロジェクトの進め方

3.1 進め方の概要

本プロジェクトでは、PDCA サイクル、スクラムなどのアジャイルソフトウェア開発の手法を学び、実践しながらアプリケーションを開発した。以下に大まかな年間スケジュールを示す。まず、5月に Git、Swift 勉強会を通じ、プロジェクトを進めるうえで欠かせない実装力を身につけた。また、同時期に第1回木古内フィールドワークも行い、開発するアプリケーションの要件定義を進めていった。6月からは本格的に開発に着手し、より詳細な要件定義やアプリケーションの設計、実装から評価、改善点の検討までの一連のサイクルを前期中に2回行った。後期も同様に開発を進め、最終成果発表の時点では1年を通して合計で4回の開発を行うことができた。

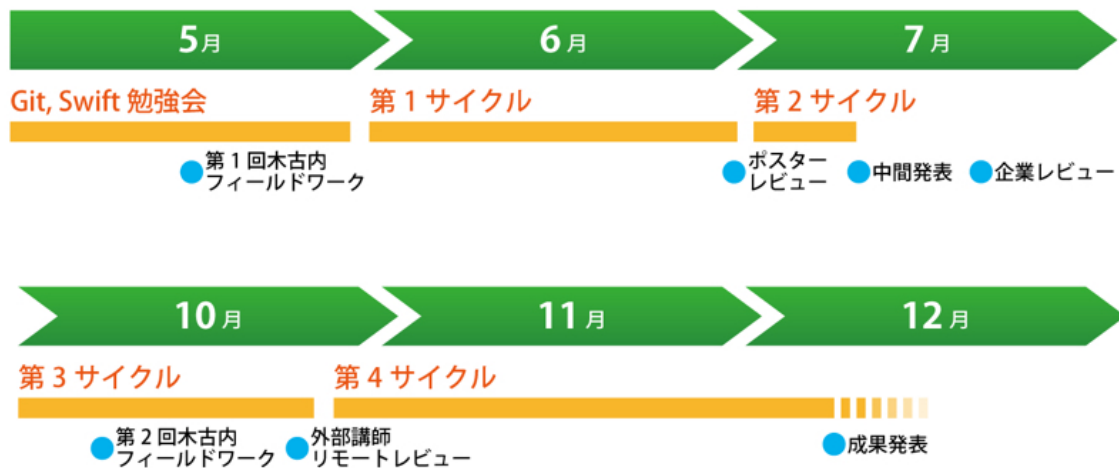


図 3.1 年間スケジュール

(※文責: 横山翔栄)

3.2 開発の進め方

PDCA サイクルの「計画、実行、評価、改善」の各フェーズについて、本プロジェクトで実践した内容について記述する。

3.2.1 計画 (Plan) フェーズ

各サイクルの始めに、どのようなアプリケーションを開発するかを計画した。計画にあたっては、開発するアプリケーションの新規性や、ターゲットユーザにもたらすメリットなどを検討し、実際に役立つアプリケーションになるよう繰り返し話し合った。以下に本プロジェクトが計画フェーズで用いたツールや手法について記述していく。

リーンキャンバス

リーンキャンバスとはビジネスモデル企画のためのツールの一つであり、新規サービスのアイデアを9種類の要素を用いて整理するためのテンプレートである。以下に要素の一覧を示す。

1. 課題、既存の代替品 (Problem, Existing Alternatives)
2. 顧客セグメント、アーリーアダプター (Customer Segments, Early Adopters)
3. 独自の価値提案、わかりやすいコンセプト (Unique Value Proposition, High-Level Concept)
4. ソリューション (Solution)
5. チャンネル (Channels)
6. 収入の流れ (Revenue Streams)
7. コスト構造 (Cost Structure)
8. 主要指標 (Key Metrics)
9. 圧倒的な優位性 (Unfair Advantage)

本プロジェクトではこの様式に則り各要素を検討することで、より綿密な要件定義を行うことができた。ただし、今回のプロジェクトにおいてはビジネス化が目的ではないため、収入の流れやコスト構造についての検討は行わなかった。

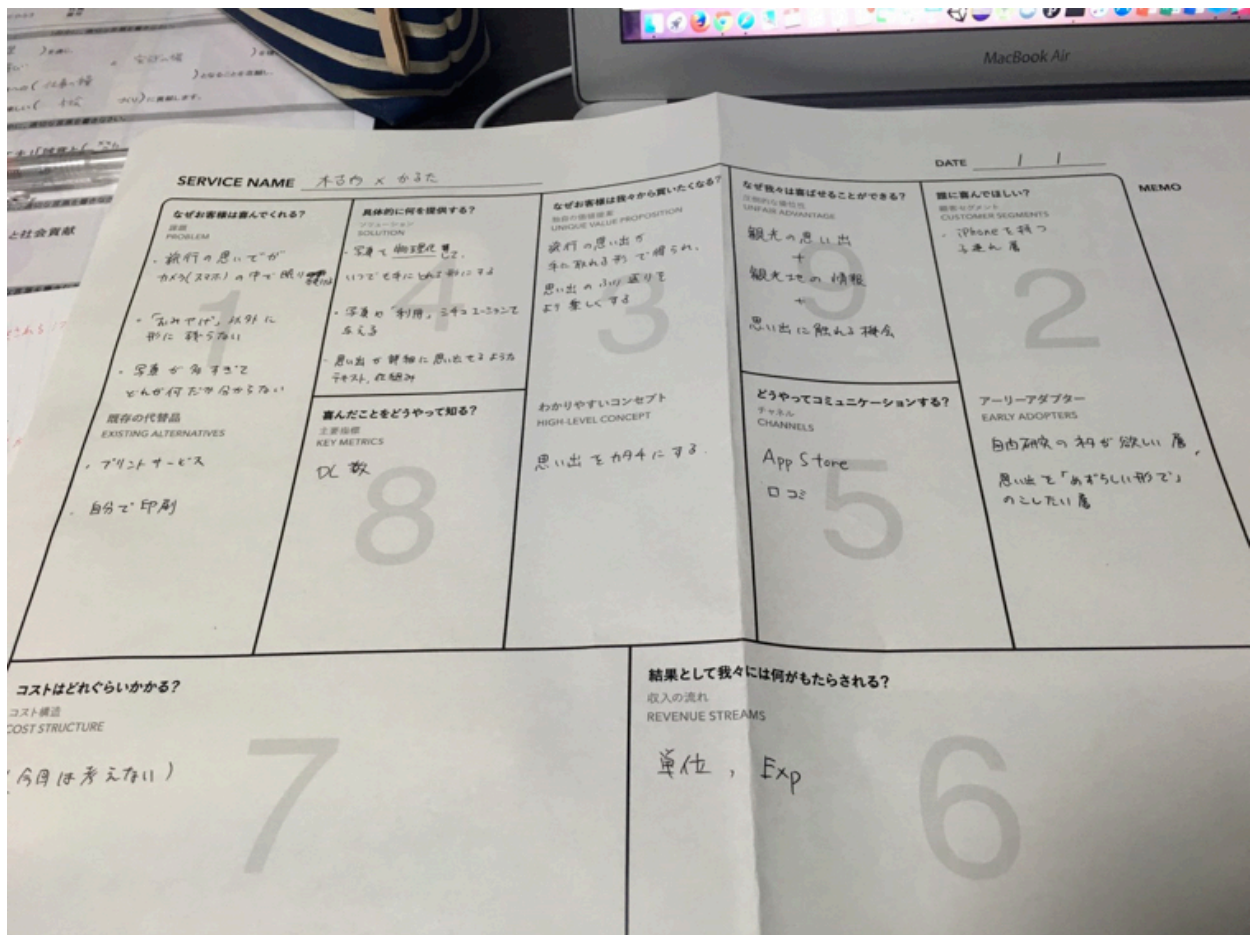


図 3.2 作成したリーンキャンバス

プロトタイプ

本格的な開発に着手する前に、簡易なモデルを作成して検討することで、問題点や改善案を挙げることができた。プロトタイピングは各サイクルでのアプリケーションの機能に関わる部分を Swift で作成したのに加え、第 3 サイクルではカルタを、第 4 サイクルではリーフレットを Adobe Illustrator で作成し実際に印刷し、検討を重ねた。実際に作成したプロトタイプについては図 5.6、図 5.7 を参照されたい。

3.2.2 実行 (Do) フェーズ

実行フェーズでは、計画フェーズで行った要件定義やアプリケーション設計をもとに、実際に開発を進めていく。以下に、本プロジェクトで実践したソフトウェア開発の手法について記述していく。

スクラム

前節でも述べたとおり、本プロジェクトではソフトウェア開発にあたりスクラムを実践した。まず、メンバのタスク管理のため次バージョンまでのタスクを列挙、細分化し、プロダクトバックログおよびスプリントバックログを作成した。本来は作業の進捗状況を毎日のスクラム会議で確認するところであるが、メンバ間のスケジュール調整の結果、毎週月、水、金曜にスクラム会議を行うこととした。スクラム会議の内容は一般的なスクラム手法と同様に、「前回の会議以降の作業報告」「次回の会議までの作業計画」「作業上の問題点の報告」である。スプリントのタイムボックスは 1 週間とし、月曜日のスクラム会議を区切りとした。また、本プロジェクトでは、タスクの進捗状況の確認や各メンバの実行中タスクの確認のため、バックログに加えかんばんも併用することとした。かんばんとは、タスクを「To-Do (残っているタスク)」「Do this week (今週行うタスク)」「In progress (メンバの誰かが現在開発中のタスク)」「Done (完了したタスク)」に分けて管理する手法である。この際、物理的なかんばんを設置する場所が確保できなかったため、Web サービスである KanbanFlow を利用した。現在のスプリントで行うべきタスクをさらにかんばんによって管理することで正確なタスク分配が可能となったほか、各メンバの進行中の作業の確認も行えるようになった。

Git 運用ルール

本プロジェクトでのアプリケーション開発にあたってソースコードのバージョン管理に GitHub を用いたが、これをトラブルなく運用するためメンバ間での取り決めを行った。まずブランチの管理に関しては git-flow モデル^{*1}を使用することとし、常に安定版および最新の開発版のソースコードを得られるよう運用した。また、ブランチのマージの際にはプルリクエストに対して 3 人以上のチェックを行うようにし、バグや動作不良が残ったまま開発が進行することを防止した。

3.2.3 評価 (Check) フェーズ

本プロジェクトではアプリケーション開発のマイルストーンを各種の発表会に設定し、発表を通じてアプリケーションに対するレビューを収集していった。発表会では実際に開発したアプリケー

^{*1} Vincent Driessen 氏が提唱した Git におけるブランチ管理規則の一つ [2]

“Swift” Application Development Based on Field Research

ションを使ってもらいながら操作性や画面遷移、使用感などの感想をもらったほか、コンセプト自体に関する意見や改善案を数多くもらえる機会となった。初期のサイクルではアプリケーションの存在意義自体を質す意見が多かったが、回を重ねるにつれコンセプトがしっかりしてきたこともあり、レビューのコメントに改善案や追加機能案が増えていった。レビューで得られた意見の中にはグループメンバが全く予想できなかったような意見もあり、改めて外部からのレビューの重要性を認識する機会となった。

3.2.4 改善 (Act) フェーズ

改善フェーズでは不十分なコンセプト設計や不足していた機能、過剰な機能の発見など、評価フェーズのレビューによって指摘されたアプリケーションの問題点を改善するための方法について検討した。この検討を踏まえ、再び計画フェーズへと移行し、要件の再定義などを行っていった。各サイクルで行った詳細な内容については5章を参照されたい。

(※文責: 横山翔栄)

第 4 章 開発準備

4.1 開発に使用したツール

iOS アプリケーションを開発する上で使用する言語は Swift を選択した。Swift とは、iOS と Mac 向けのアプリケーションを開発するために Apple が提供しているプログラミング言語である。開発したソースコードのバージョン管理ツールとして、Git/GitHub を使用した。Git はソースコードやファイルの変更履歴を管理することができるシステムである。Git は変更履歴をリポジトリと呼ばれる場所に保存する。リポジトリには各メンバーの開発環境に作成するローカルリポジトリと、メンバーで変更履歴を共有するリモートリポジトリの 2 種類が存在するが、GitHub はこのうちリモートリポジトリを提供するサービスの一つである。各メンバーの作業内容、すなわちソースコードの変更履歴をリモートリポジトリで共有、統合することができるため、複数のメンバーが同時に開発を進めることができる。タスク管理には、KanbanFlow という Web サービスを使用した。KanbanFlow はオンライン上で利用できる無料のタスク管理システムである。KanbanFlow はやらなくてはならないタスクを作業予定、作業中、作業完了の 3 つの段階に分ける。また、そのタスクそれぞれに担当者を記入する。これにより、メンバーのタスクが今どのような状態なのかをチームで共有することが可能になった。

(※文責: 池田俊輝)

4.2 勉強会

Swift と Git/GitHub の勉強会を 5 月に週に 1 回、1 ヶ月かけて行った。この勉強会では TA の方々から指導していただいた。進め方として 1 時間ほど TA がスライドを用いて講義を行い、その後 2 時間程度演習形式で TA に用意してもらった例題を解いた。具体的にはアプリ内でマップを表示し、自分の指定した場所にピン型のマーカーを設置し、その場所をタップすると自分の指定した Web サイトに遷移するアプリを作る演習を行った。勉強会中にわからないところがあれば、TA に個別に対応していただいた。アプリ開発の勉強会では Xcode という iPhone アプリケーション開発用デベロッパーツールの基本的な使い方を学んだ。そこでストーリーボードの使い方や、デバッグするために Mac 上で iPhone や iPad の環境をシミュレートする方法などを学んだ。

Git/GitHub の勉強会ではバージョン管理システムの目的や、管理方法の基礎を学んだ。Swift と Git/GitHub を使用した開発の練習を目的として、サンプルアプリの開発を行った。実際にプログラムの構造を考えてどの機能の実装を誰が行うのか決めて、ソースコードのファイルを共有しながら各自がコーディングを行うといった全体の流れを確認した。

(※文責: 池田俊輝)

第 5 章 開発プロセス

5.1 第 1 サイクル (5 月 1 日～7 月 3 日)

第 1 回フィールドワーク

我々は木古内町の観光産業と開発するアプリの方向性を考えるにあたり、木古内町観光の解決すべき問題点の発見を目的として木古内町でフィールドワークを行った。このフィールドワークでは、多角的に木古内町を観察するために函館市-木古内町間の交通手段をメンバー間で電車・自転車・自動車の 3 つに分担し、それぞれが観光者の視点に立って調査を行った。木古内町でのフィールドワークを行って気づいたことは主に 2 点ある。まず 1 点目は、木古内町の観光情報が分散して存在していることである。そのため、木古内町に着いた際、どこで何ができるのか分かりにくかった。これは、前述の木古内町観光協会 Web サイトや SNS、パンフレットなど多くのメディアで情報が発信されていることで観光情報が分散していて観光者に伝わっていないことや、情報提供するコンテンツが木古内の魅力的な部分を十分に伝えられていないことが原因だと考えられる。気づいたことの 2 点目は、観光を終えた人々には観光の体験や思い出を友人などの親しい人に伝えたいというモチベーションがあるという点である。観光の体験や思い出を伝えるという行為には、その相手に観光地の良さを伝え、新たな観光客を呼び込む手がかりとなる要素を含んでおり、それをより魅力的に伝えるには写真を使うのが効果的である。しかしながら、多くの観光者はその写真の整理をあまり行っておらずそれを効果的に伝えられていない。以上の気づきを元に第 1 サイクルの開発を開始した。

(※文責: 細川椋太)

第 1 サイクルはプロジェクトが発足した 5 月 1 日から 7 月 3 日に行われた中間発表会用のポスターレビューまでとした。本サイクルでは第 1 回フィールドワークの経験から、木古内町の観光情報が分散していること、観光中に撮影した写真が整理されていないことを問題として定義した。最初の設計段階であがった案は以下の 5 つであり具体的な画面イメージを作成できたのはマップ画面、Web ページを表示する画面、お散歩コースを表示する画面の 3 つである。これらの案に対するレビューを 6 月 12 日に企業講師の方から受けた結果、現在の提案に木古内らしさを加える必要があるとの指摘を受けた。

初期段階でのアプリ機能案

- マップ上に表示されたピンをタップすると吹き出しを表示し、吹き出し内に観光スポット名やお店の営業時間および連絡先を表示する機能
- 吹き出しをタップするとお店の Web ページへ画面遷移する機能
- 木古内町民しか知らないニュースや、木古内町の天気をマップ画面に表示する機能
- ユーザ同士で木古内町の写真を共有できる機能
- 木古内町観光協会が公開している木古内お散歩コースをマップの中に盛り込む

以下に示す図 5.1 は作成した画面のイメージである。図 5.1(a) は、マップ上にピンが自動的に配置され、そのピンをタップした時に表示する画面のイメージである。この画面では観光スポット

を検索できたり木古内町の天気を表示したり、表示するピンのカテゴリを切り替えることができる。図 5.1(b) は、図 5.1(a) に表示している吹き出しをタップすると画面遷移して表示する画面のイメージである。この画面には、観光スポットの詳細情報としてお店のメニューや営業時間、商品の写真などを記載する。図 5.1(c) は、木古内町観光協会が公開しているお散歩コースを表示している画面イメージである。画面下のタブをタップして選択すると表示するコース内容に対応するコース内容へ切り替えることができる。また、画面上にはお散歩コースの途中にある観光スポットを赤点で表示しており、その赤点をタップすると図 5.1(a) と同様に吹き出しと対応する情報を表示する機能が備わっている。



図 5.1 初期段階での画面イメージ

その後は上記の初期段階でのアプリ機能案に対して議論を行い、木古内町民しか知らないニュースは継続運用が困難であるという理由から実装を保留した。また、木古内町観光協会が公開している木古内お散歩コースをマップの中に盛り込む機能も実装を保留した。理由は、公開されていたコース内容が車での移動を前提にしており、車を運転しながらスマートフォンを見ることになるため同伴者が必要という縛りを設けてしまうからである。この段階で実装した機能は下記の4つである。1つ目は、マップ上に表示されたピンをタップすると吹き出しを表示し、吹き出し内に観光スポット名やお店の名前および連絡先を表示する機能である(図 5.2(a) 参照)。2つ目はカテゴリごとにピンの表示を切り替える機能である(図 5.2(a) 参照)。3つ目は、現在地から目的地までのルート案内を行う機能である(図 5.2(b) 参照)。4つ目は、木古内町の天気を表示する機能である(図 5.2(c) 参照)。上記4つの機能に対してチーム内で中間発表会に展示する内容として不足はないと判断したが、中間発表会用のポスターレビューの際にユーザストーリーに問題があるとの指摘を受けた。指摘の内容は、観光客は観光スポットやランドマークの写真を見てからその場所へ行くのが通常の流れであるが、当時作成していたアプリは場所を伝えてから写真や詳細情報を見せる流れになっており、情報を提供する順番が逆である。この状態だと木古内町の魅力はもちろんのこと、何があるかすら伝わらないとの内容であった。この指摘を受けて先に伝える情報は場所ではなく内容にすべきであるという結論に達した。そして中間発表会までに改善することをチーム内で合意し、

第2サイクルへ移行した。

最後に、本サイクルで実装した4つの機能を実行している時の画面を図5.2に示し、それらについて詳細を述べる。1つ目の機能と2つ目の機能を実行している画面は図5.2(a)である。この画面はユーザが一番最初に見る画面である。この画面ではマップ上に表示されたピンタップすることで、観光スポットの名前またはお店の名前と連絡先が記載してある吹き出しを表示することができる。また、画面下に表示している食べる、見る、買うの3つのカテゴリボタンをタップすることで対応するカテゴリのピンをマップ上に表示することができる。図5.2(b)は3つ目の機能を実行している時の画面である。図5.2(a)中にある吹き出し右側に表示している青いアイコン*i*をタップすることで現在地から対応する場所までのルートが赤線で表示される。なお、図5.2(b)は木古内駅を現在地として設定している状態である。図5.2(c)は、木古内町の天気を表示している画面である。現在の最高気温および最低気温、3時間ごとの天気、1週間の天気予報を表示している。天気の表示に使用した技術については付録A.1を参照されたい。



図 5.2 第1サイクルでの完成画面

(※文責: 岩見建汰)

5.2 第2サイクル(7月3日~7月10日)

第2サイクルは中間発表会用のポスターレビューが終わった日から7月10日に行われた中間発表会までとした。本サイクルでは第1サイクルで課題として残ったユーザストーリーの流れを改善すること、第1回フィールドワークであげられた観光中に撮影した写真が整理されていないという課題を解決すべき問題として定義した。本サイクルでは2つの問題に同時に取り組んでいった。

1つ目のユーザストーリーの流れを改善することに対しては、中間成果発表会まで残り1週間で切っていたため新たな機能を追加して修正するのは間に合わないと判断し、第1サイクルで実装していた部分の情報の見せ方を改善する方針で議論を進めた。具体的に修正した内容を表5.1に、実装した画面は図5.3に示す。2つ目の観光中に撮影した写真が整理されていないという問題への解決アプローチとしてフォトストーリーという機能を考案した。これはユーザが撮影した写真と移動

“Swift” Application Development Based on Field Research

した経路を表示したマップが図 5.4 のような紀行 *¹ となり、木古内町観光をよりリアルに振り返れる仕組みである。当時考案していたフォトストーリーのユーザストーリーの流れとフォトストーリーに対するレビュー内容を以下に示す。

なお、本サイクルではアプリ全体のユーザストーリーの流れは多少改善されたが、他にも改善すべき点が多く残る状態となった。フォトストーリーに関しては、ユーザが写真を撮るという行為にワクワクする付加価値 *² が必要という課題が残った。

表 5.1 第 1 サイクルから第 2 サイクルへの変化

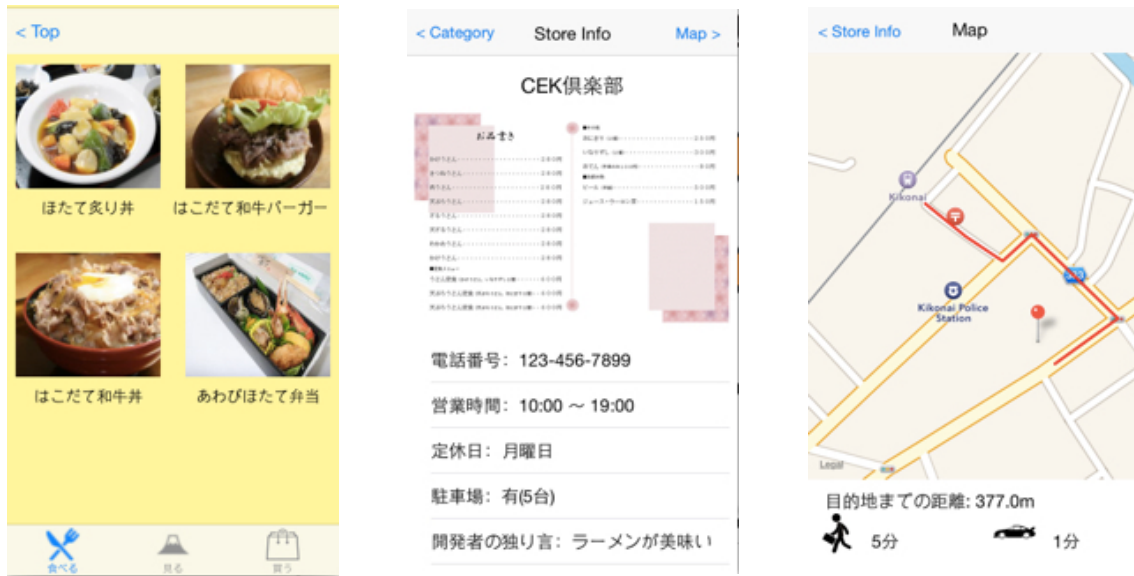
改正前	改正後
食べる・見る・買うのカテゴリ別にピンを表示	カテゴリは変えずに写真の一覧を表示
お店の詳細情報として Web ページを表示	詳細情報の表示方法は我々で作成した画面構成を用いる
マップ画面を最初に表示	カテゴリ別になった写真一覧を最初に表示
目的地までのルートのみ表示	ルートの他に距離と徒歩及び車での所要時間を表示

画面の詳細説明 (図 5.3)

- 図 5.3(a) は、第 1 サイクルで問題となった内容よりも場所を先に伝えてしまっていたことを改善した画面である。この画面ではカテゴリにアイコンを追加し、マップではなく食べ物やスポットに関する写真を一覧表示している。各写真をタップすると図 5.3(b) へ画面遷移する。
- 図 5.3(b) は、スポットの詳細情報を表示する画面である。お店を例としているため、メニュー表や連絡先、営業時間の他にも駐車場や開発者の独り言をサンプル表示している。
- 図 5.3(c) は、図 5.3(b) の画面右上に表示している「Map >」をタップすると現在地から対応するスポットまでのルートを表示する画面である。第 1 サイクルでのルート案内に加えて目的地までの距離と所要時間を表示してある。

*¹ 旅行中の出来事、行動、見聞きした事、感想などを書いたもの

*² ユーザを能動的にさせる仕組みのことである。例えば、撮影した写真の枚数に比例してクーポン券がもらえるなどである。



(a) 観光スポットの紹介

(b) 詳細情報の表示

(c) ルート案内

図 5.3 第 2 サイクルの完成画面

フォトストーリーのユーザストーリー

1. フォトストーリーを使用することでどのような紀行を作成できるのかを説明するサンプル動画を見る。この動画にはフォトストーリーの操作説明及び木古内町の魅力が内容として含まれているため、操作方法だけでなく木古内の魅力も知ることができる。
2. 観光中に撮影した写真をアプリ内に保存する。
3. 撮影した写真の一覧が表示され、ユーザが写真を任意のカテゴリ別に分けることができる。
4. 加工ボタンをタップするとユーザが撮影した写真と移動した経路を表示したマップが紀行として画面に表示される。
5. フォトストーリーを使用してユーザの思い出話によるコミュニケーションが促進される。

フォトストーリーに関する意見

- 木古内町の歴史に関する情報を表示する機能が欲しい。
- 図 5.4 に対して、1つの画面に写真が2枚もあると分かりにくい。
- 写真を撮りたくなるトリガーがないため、フォトストーリーの存在が危うい。
- 時系列での説明をされると聞いている側は辛くなってしまう。2,3枚ほどのダイジェストの方が良い。
- フォトストーリーを実装するのであれば、誰が写真を撮るのかを考えて UI 設計を行うこと。スマートフォンに使い慣れた若い人と大人と子供ではそれぞれ UI が違ってくる。
- 案内だけに留まらずに思い出に着眼したことは良い。今後の掘り下げ方で何倍にも良いものになる可能性がある。

(※文責: 岩見建汰)



図 5.4 紀行の画面イメージ

5.3 第 3 サイクル (9 月 25 日～10 月 21 日)

第 3 サイクルは後期が始まった 9 月 25 日から 10 月 21 日に行われた企業講師とのテレビ会議までとした。本サイクルでは、第 2 サイクルで見つかった大きな課題の一つである写真を撮る行為にもっとワクワクするような付加価値を加えなければならないこと、そして観光情報をより魅力的に紹介することを課題として活動を進めた。

まず我々は、後期が始まって最初の講義で前期を振り返り、もう一度課題を見直した。そして観光情報の表示の仕方について話し合ったとき、Time Out Tokyo[3] が掲載していた「札幌でしかない 50 のこと」という記事の存在を知り、それを参考にして木古内で「できること」に着目した情報を表示する形式にすることを決定した。また、それと同時に「アプリからランプを作ることができたら面白いのではないか」という意見が出たため、そのアイデアを膨らませて、撮った写真を利用してカルタという「もの」にする機能を作ることも決定した。カルタという「もの」にする意義は後に説明する。また、この時、テレビ会議までにはできる範囲までは実装を行おうと決め、木古内で「できること」に着目した情報紹介の機能から実装を進めた。具体的に修正した内容を表 5.2 に、完成した画面イメージと印刷して完成したカルタのサンプルを図 5.5、5.6 に示す。図の 5.5 は (a) がメインとなる画面で、この画面から (b) や (c) に遷移することを想定して設計・実装した。それと同時に、アプリ内で紹介する観光スポットの写真を集めるために 10 月 15 日にメンバでもう一度木古内へ行き、アプリ内で使用する写真の素材を集めた。詳細は次頁で説明する。パンフレットや Web で掲載されていないような自分たちが見つけた魅力のあるスポットの写真も撮影できた。さらに、本サイクルでまだ決めていなかったアプリのタイトルをメンバ間で話し合った結果、アプリ名を「キーコ紀行」と正式に決定した。

本サイクルではより魅力的に観光情報が表示でき、写真を撮る動機づけとなるアイデアを提案できた。しかし、カルタは 50 枚を超える紙を印刷しなければいけないため手間がかかるという課題が残った。また、技術面としてはメンバに割り振った部分が独立しており、アプリが部分的にしかできていない状態だったため、早急にメンバ間で開発したものを結合する必要があるがあった。アプリを当時の直近のイベントであった HAKODTE アカデミックリンク 2015 (以降、アカデミックリンクとする) に開発を間に合わせるようにスケジュールを立てることを決定した。

表 5.2 第 2 サイクルから第 3 サイクルへの変化

改正前	改正後
カテゴリごとに写真の一覧を表示	写真と同時に紹介文を加えることで木古内で「できること」に着目した情報を表示
詳細情報を表示してからマップに遷移	マップ画面と同時に詳細情報や写真を表示
「フォトストーリー」という機能でアプリ内で写真を振り返る	カルタという「もの」にして思い出を残す

第 2 回フィールドワーク

このフィールドワークは 2015 年 10 月 15 日に木古内町で行った。フィールドワークの目的は 2 点で、1 点目が「木古内町でできること」を探り発見することであった。この「木古内町でできること」のフォーマットは、ある場所でできることに対して写真と説明文のセットとした。2 点目は、その「できること」に対応する紹介写真の撮影であった。目的が明確だったため、第 1 回フィールドワークのようにメンバごとに交通手段を分担するというではなく、また効率的に情報を収集するために、2 つのグループに分かれて行動した。このフィールドワークを行い、計 15 個の「木古内町でできること」のコンテンツを作成した。実際に現地調査を行ってできることを探ったため、コンテンツにフィールドワークでの実際の体験を「何ができるか」という形で反映することができた。特に、旧 JR 渡島鶴岡駅跡のように観光スポットとしてはあまり有名でなく、Web 上でも観光パンフレットでも確認できなかった場所を発見し、紹介できるようになったという点は、現地調査を行った利点を上手く活かした例であった。現地調査からの情報収集は問題なく行えたものの、紹介写真の撮影は写真にグループメンバが映り込んでしまっているなど、不十分な部分もあったため、その部分は反省すべき点であった。



図 5.5 第 3 サイクルの完成画面イメージ

“Swift” Application Development Based on Field Research

カルタにする意義

- iPhone という小さい画面の中で振り返る必要が無い。
- 写真だけでなく、言葉と一セットになる形式を持っているため、より強く思い出を振り返ることができる。
- 実際に「もの」という形にすることで長く保管されやすい。
- 遊びながら思い出を振り返ることができる。

本サイクルの提案に対するレビュー内容

- 「できること」に着目したことによって「どこに行こう?」から「ここに行きたい!」に近づいた。
- 実際に印刷する段階まで行ってほしい。
- 類似サービス (例えば富士フィルム株式会社のフォトブック [4]) の観光版のフォトブックとして作ってみるのは良い。
- ニーズはあると思う。
- 図 5.6 のようなカルタとは違う別の何かにした方が良い。手間がかかるから。
- 写真のプロトタイプを作って木古内の関係者に提案するのも良い。まずは見てもらうことが重要。



図 5.6 カルタの完成サンプル

(※文責: 山川拓也)

5.4 第4サイクル（10月22日～12月11日）

第4サイクルは企業講師とのテレビ会議が終わった次の日の10月22日から12月11日の成果発表会までとした。本サイクルでは、第3サイクルで進めていた実装をさらに進めること、そしてカルタに代わる何か手間のかからない「もの」を提案することを課題として活動を行った。

カルタ以外で手間がかからないものは何かを議論した結果、リーフレット^{*3}という案が生まれた。カルタのように50枚分の印刷をしなければいけないような数による手間をかけず、さらに一枚の紙で作成できて低コストに抑えられるため、この案を採用した。リーフレットの表側には従来の観光パンフレットと同様に木古内の名産などの基本的な観光情報を記し、裏側にはユーザが撮影した写真を自動的に配置して独自性のあるものを記せるようにデザイン・実装することにした。具体的な完成イメージは次章の6.5で説明する。必ずしもユーザがリーフレットを印刷するとも限らないため、アプリ内でも振り返ることができるようにアルバム機能を作ることを決定した。本サイクルで実装した機能は大きく分けて、観光する機能、振り返る機能、そして印刷する機能の3つだ。11月14日にアカデミックリンクがあったため、一旦そこをマイルストーンとして開発し、多くの企業の方や一般の方からレビューを受けた。そのレビューを受けて成果発表会までさらに開発を続けた。具体的に実装した機能・画面は次章で詳説する。前回のサイクルと本サイクルでの主な変化を表5.3に示す。アカデミックリンクと成果発表会でのレビュー内容を後に記述する。

本サイクルでは長く続けてきた要件定義が十分に固まったため、比較的実装に集中できて完成度の高いアプリを開発できた。最終報告会での評価も高かったため、今後はプロジェクトが終わっても開発を続けてリリースすることを決定した。そのために、今後は木古内の関係者の方々にキーコ紀行を実際に提案しに行き現地の方のレビューを受けてさらに質の高いアプリにすること、現在残っているバグを取り除くこと、そしてどのように運営していくかを決定することが当面の課題となる。

表5.3 第3サイクルから第4サイクルへの変化

改正前	改正後
カルタを作る機能	リーフレットを自動生成する機能
カルタから思い出を振り返る	アルバム機能または、リーフレットを用いて思い出を振り返る

アカデミックリンクでのレビュー内容

- 継続的に使ってもらう仕組みが足りていない。
- フォントが統一されていない。
- リーフレットの写真をもう少しまとめてほしい。
- ボタンのアイコンや色の配色など、デザイン面をもう少し改善してほしい。

成果発表会でのレビュー内容

- 利便性とカスタマイズできる点が面白い。
- アプリをどうやって広めていくのかがあまりわからなかった。
- 自分だけのリーフレットを作れるのは面白い。

^{*3} 一枚刷りの印刷物、折りたたみ式の小型の印刷物 [5]

“Swift” Application Development Based on Field Research

- 拡張の可能性を感じる。
- もっと木古内らしさを出してほしい。
- リーフレット作ってもゴミになってしまうのでは。
- 操作説明が欲しい。
- 検索機能が欲しい。
- データではなくものにできるのは良い。
- UI デザインが分かりやすい。
- 観光情報が分かりやすくまとまっている。
- 写真が無駄にならなくて済む。

(※文責: 山川拓也)

第 6 章 キーコ紀行について

6.1 キーコ紀行の概要

本プロジェクトで開発した iOS アプリケーション「キーコ紀行」は、「木古内町でできること」に着目した観光情報の利用と、観光スポットで撮影した写真を用いてオリジナルのリーフレットの作成ができるアプリケーションである。このアプリケーションの名称は、「木古内町」という地域の名称と、旅行の体験を記録した文章を指す「紀行」を組み合わせたものである。「キーコ紀行」はユーザに観光の思い出を残してもらうことを目標としており、使用場面は木古内来訪前の観光プラン作成から観光後の印刷までを想定している。既存の他の観光アプリケーションと比較した際の優位性としては、「キーコ紀行」は観光情報の提供だけでなく、思い出をアプリケーションの中にまとめてパッケージとして保存できる「アルバム機能」を有する点や、撮影した写真を利用してオリジナルリーフレットを印刷でき、思い出を形にして残すことができる点などがある。次節より「キーコ紀行」の各機能について詳しく記述していく。

(※文責: 横山翔栄)

6.2 「観光する」機能

6.2.1 カードリスト画面

カードリスト画面では、木古内で「できること」に着目した観光スポットを写真と紹介文で「カード」という形式にして、それをリスト表示する。一枚のカードに含まれる情報は、そのスポットの写真、キャッチコピーの様なタイトル、紹介文、そしてテキスト編集画面に遷移するボタン、マップ・詳細画面へ遷移するボタン、写真撮影画面へ遷移するボタン、そしてお気に入りボタンといういわゆるブックマーク機能を持つ 4 つのボタンである。詳しくは表 6.1 を参照されたい。そしてアプリの画面を図 6.1 で紹介する。ユーザはキーコ紀行をダウンロードしたらまずこの画面を見て、気になったらそのスポットへ向かう、といったユーザストーリーを想定している。使用されている写真は、実際に我々が木古内に行き、撮影したものを使用している。紹介文は、メンバで考えた約 70 字という制限がついた文である。仮に 70 字以上の文をカードに表示した場合、カードの見栄えが悪くなってしまうため、見栄えを悪くさせないようにすることを目的として制限した。この画面からは、マップ・詳細画面、写真撮影画面、テキスト編集画面に遷移できる。

表 6.1 カードリストのボタンのアイコンとその意味





アイコン	意味
	テキスト編集画面へ遷移するボタン。
	マップ・詳細画面へ遷移するボタン。
	写真撮影画面へ遷移するボタン。
	いわゆる「ブックマーク」のようなボタン。ユーザが気に入ったらこのボタンを押し、マップ詳細画面で違うスポットを見たときにでも場所がわかるようになる。



図 6.1 カードリスト画面

(※文責: 山川拓也)

6.2.2 マップ・詳細画面




マップ・詳細画面は、ツールバー、マップ、カードから構成されている。ツールバーはカードリスト画面へ戻るために画面上部に表示している。マップはユーザの現在地や観光スポットの場所を見るために画面中央に表示している。カードは観光スポットの場所とそこに関する情報を関連づけて見ることが可能とするために、観光スポットのタイトルと写真および紹介文を画面下部に表示している。

本画面へ遷移してきた際の初期動作は、カードリスト画面でユーザがタップしたカードに対応する観光スポットの場所をピンとしてマップ中央に表示することである。また、カードリスト画面でユーザがお気に入りとして星をつけた場合は対応する観光スポットの場所を星としてマップ上に表示する。マップ上に表示しているピンまたは星をタップすることで、画面下部のカードに記載している情報が対応する観光スポットの情報へと切り替わる仕様である。初期状態では、画面下部の

“Swift” Application Development Based on Field Research

カードの情報はお店の営業時間や有名商品の値段などを表示 (図 6.2(a)) しているが、右向きの矢印をタップすることでカードリスト画面にも表示している観光スポットの紹介文へと表示内容が切り替わる (図 6.2(b))。観光スポットの紹介文を表示している間は左向きの矢印となり、それをタップすることで初期状態の詳細情報の表示へと切り替わる。マップ・詳細画面で使用したアイコンの意味については表 6.2 を参照されたい。

表 6.2 アイコン一覧と意味

アイコン	意味
	本画面への画面遷移直前にタップした観光スポットの場所を示すピン。
	カードリスト画面で星ボタンを押したカードの観光スポットの場所を示すピン。
	観光スポットの詳細情報と紹介文の表示を切り替えるためのボタン。



(a) 観光スポットの詳細情報を表示中 (b) 観光スポットの紹介文を表示中

図 6.2 マップ・詳細画面

(※文責: 岩見建汰)

6.2.3 写真撮影画面

写真撮影画面では iPhone に内蔵されているカメラを使用して観光スポットの写真を撮影する。本画面の意図は、カードリスト画面で木古内町の魅力を知り、観光スポットを訪れたユーザにそこを訪れた記念を残してもらうことである。本画面へは、カードリスト画面のカメラボタンをタップすることで遷移する。遷移してきた直後の画面を図 6.3(a) で示す。写真の撮影は、画面の下中央にある丸いボタンをタップして行う。撮影時は左上のボタンによりフラッシュの有無を選択できる。右上のボタンでは、インカメラとアウトカメラの切り替えをすることができる。また、左下の「キャンセル」ボタンをタップすることで本画面からカードリスト画面へ戻ることができる。

撮影後は、撮影した写真を保存するか再撮影を行うかを選択する画面へ遷移する。撮影後の画面を図 6.3(b) で示す。右下の「写真を使用」 ボタンをタップすることで写真の保存が行える。保存した場合は、撮った場所に対応するカードに上書きされ、表示される。また、写真を撮り直す場合は、左下の「再撮影」 ボタンをタップすることで撮影時の画面に戻ることができる。



図 6.3 写真を撮影する機能の画面

(※文責: 池田俊輝)

6.2.4 テキスト編集画面

テキスト編集画面は観光スポットに対応するカードに表示する紹介文をユーザが編集する画面で、編集結果は前述のカードリスト画面やマップ・詳細画面、後述の「振り返る」機能、印刷されるリーフレットに反映される。編集できる項目はタイトルと本文の2つである。この編集機能は、もともと観光スポットの説明が書かれているものを、ユーザがその場所で体験したことや起こったことに関するエピソードに置き換えることによって、思い出をより印象深いものとして残す効果や、読み返したときの振り返りをより深いものにするという効果を期待して実装した。画面は図 6.4 のような構成になっており、白い欄内の文字を書き替えて「保存する」ボタンをタップするとその結果が各画面に反映される。レイアウトは極力カードリストで表示されるものと同じようにして、ユーザが何を編集しているかが分かるよう工夫した。またタイトルと本文がそれぞれ編集可能であることを示すためにテキストエリアの背景色を白とし、この画面へ画面遷移した際に予めテキストエリアが選択され画面下にキーボードが表示される状態にした。ほかに、担当教員からの、ユーザが操作を誤ってもそれを取り消せる仕組みが必要との助言から「元に戻す」ボタンを配置した。これはユーザが誤って保存の操作をしてしまった場合の救済措置で、このボタンをタップすることで文章を最初の状態に戻すことができる。また、ユーザの誤った操作を未然に防ぐための対策として、確認ダイアログを表示するようにした。テキストを変更して保存せずに前の画面に戻ろう

とした場合や、元に戻すボタンをタップした際に、確認ダイアログで再度ユーザに確認を求めるステップを設けて誤った操作を減らす。



図 6.4 テキスト編集画面

(※文責: 細川 椋太)

6.3 「振り返る」機能

振り返る機能は観光中にユーザが撮影した写真もしくはデフォルトでアプリ内に入っている写真に対応するスポットの紹介文と共にスワイプ操作で見ることができる機能である (図 6.5(a))。撮影した写真とデフォルトの写真の表示は、画面右上にあるタブをタップすることで切り替えられる。また、ユーザが撮影した写真が1枚もない場合も考えられるため no image などの表示ではなく、木古内町マスコットキャラクター「キーコ」がお辞儀をしている画像と、写真を撮ると表示される旨のメッセージをカードリスト画面の雰囲気と統一をさせて表示した (図 6.5(b))。

本機能には、スワイプできることが伝わらないということや全部で何枚の写真があるのかが分からないという問題が現状では存在する。それらに対して、矢印を左右に表示したり pagecontrol という図 6.6 のような何個中何個目という情報を表示できる部品を活用する案があった。しかし、画面上にスワイプ操作ができるという矢印を表示した場合の画面レイアウトを期間中に考案することが困難だったため、現状のままとしている。また、pagecontrol については写真の枚数が膨大になることで、何個中何個目という情報の表示を画面に収めることが困難となり、アプリの品質を落とす可能性があるため実装を行わなかった。



(a) 振り返る機能の表示例

(b) 撮影写真なしの表示例

図 6.5 振り返る機能の画面

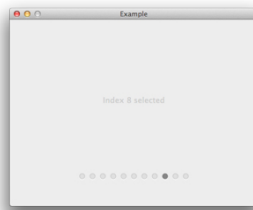


図 6.6 pagecontrol のサンプル

(※文責: 岩見建汰)

6.4 「印刷する」機能

印刷する機能はユーザがカードリストから使用したい写真を選択することで、1枚のリーフレットを自動で作成する機能である。本機能は、写真選択画面、リーフレット作成画面、印刷プレビュー画面、AirPrintの設定画面の4つの画面から成り立つ。

ユーザが最初に見る画面は写真選択画面である。この画面へはトップ画面の「印刷する」ボタンをタップすることで遷移する。遷移したときの画面を図 6.7(a) に示す。この画面はリーフレットと縦横比が同一になっており、横スクロールすることでリーフレットの全体を見ることができ。リーフレットの詳細については6章5節を参照されたい。また、写真撮影画面内に表示している「タップして写真をえらぶ」と書かれている枠内をタップするとリーフレット作成画面に遷移す

“Swift” Application Development Based on Field Research

る。このときの画面を図 6.7(b) に示す。リーフレット作成画面ではリーフレットで使いたい写真をユーザに選んでもらう。表示された写真の中にユーザが使用したい写真があれば「この写真を使う」ボタンをタップすることで写真選択画面に遷移し、「タップして写真をえらぶ」と書かれている枠内が選択した写真に変更される。このようにしてリーフレットに使いたい写真を 7 枚選んでもらう。リーフレットに使う写真が 7 枚であるため、7 枚選んでいない状態で写真選択画面の右上に表示している「完了」ボタンをタップすると警告が表示される。警告は全ての枠に対して写真を選択するように注意する内容である。ユーザが 7 枚全てを選んだ状態で「完了」ボタンをタップすると印刷プレビュー画面に遷移する。このときの画面を図 6.7(c) に示す。印刷プレビュー画面では、リーフレットの表側、裏側がどのように印刷されるのかを確認することができる。もし、印刷されるリーフレットを修正したい場合は左上の「戻る」ボタンをタップして写真選択画面に戻ることができる。印刷プレビューに表示しているリーフレットを印刷する場合は右上の「印刷」ボタンを押すことで AirPrint^{*1} の設定画面に遷移する。このときの画面を図 6.6(d) に示す。AirPrint の設定画面ではまず、接続するプリンタを選択する。iPhone とプリンタが Wi-Fi に接続されている場合はプリンタの欄に使用可能なプリンタ名が表示される。次に、印刷する部数および印刷するページを片面か両面かを選択した後、右上の「プリント」ボタンをタップすると iPhone からプリンタにデータが送信される。そして、プリンタが Wi-Fi 経由でデータを受信するとリーフレットが印刷される。

*1 AirPrint は Apple Inc. の商標



(a) 写真選択画面



(b) リーフレット作成画面



(c) 印刷プレビュー画面



(d) AirPrint の設定画面

図 6.7 印刷機能の画面

(※文責: 池田俊輝)

6.5 リーフレットについて

このリーフレットはユーザの木古内町の観光の思い出を1枚の紙面にまとめたものである。リーフレットという形は、ユーザの手間を極力減らすという制約のもと考慮し、紙1枚を両面印刷するだけだが存在感があるという点や、観光の思い出としてその場所のリーフレットやパンフレットを保管する人が一定数いるという背景から決定した。このリーフレットの目的は二つあり、一つは、ユーザが撮影した写真を手がかりにしてその思い出を振り返ることで、もう一つは、木古内町観光をした人がこのリーフレットを他の人に見せることによって木古内町の認知度を向上させ、観光客の増加を促進させることである。リーフレットはキーコ紀行を使用して木古内町観光を行い、そのあとで7枚の写真を選択することで家庭のAirPrint対応のプリンタを用いて印刷可能である。用紙サイズは、家庭で新たに用紙を購入することを避けるために使用される頻度の高いA4とし、折り方は巻3つ折りとした。リーフレットの表面(図6.8(a))の表紙部分には木古内町をイメージさせる、海・山・木古内市街地・新幹線の駅といった木古内町の要素が多く詰まった写真に、木古内町観光をしたときのものと分かるように、「みそぎの郷きこない」の文字を表記した。裏表紙や表紙を開いた面には、木古内町の位置を示す地図や基本的な情報、特産品を記し、木古内町の認知度向上やリピーターの増加を狙った。リーフレットをすべて開くと、A4紙全体にユーザが撮影した7枚の写真とその場所に関する説明の文章が印刷された面が見える(図6.8(b))。この部分がリーフレットの一番の目的で、ここを見ながらユーザに思い出を振り返ってもらうことを想定している。ここに印刷される文章は、アプリ内でもともと設定されているものであるが、よりオリジナリティを求めるユーザは、前述の編集機能を用いて文章を変更することができる。ユーザがその場所で体験したことや起こったことに関するエピソードを記せば、その文章と写真がセットで見られることによって思い出の振り返りがより深いものになり、記憶に残るものとなることを期待している。

(※文責: 細川椋太)



(a) 表面。この面の情報は固定。

図 6.8 印刷されるリーフレット



(b) 裏面。ユーザの撮影した写真が印刷される。

図 6.9 印刷されるリーフレット

第 7 章 使用技術

7.1 データベース

このアプリでは、モバイルデバイス向けのデータベースである Realm を使用した [6]。これはサーバ上ではなくデバイスの内部メモリにデータベースファイルを作成し、そのファイルで様々なデータを扱うものである。このデータベースファイルの内容を閲覧するためには Realm Browser を用いた。このデータベースを使用するにあたり、重要な部分のコードを以下に記述・解説する。

Realm モデルオブジェクトの定義

```
class CardData: Object {
    dynamic var ID = 0
    dynamic var cardText: CardText?...

    dynamic var updated = false

    override class func primaryKey() -> String {
        return "ID"
    }
}
```

モデルオブジェクトはクラスで宣言して定義する。このテーブルのカラム要素は “dynamic var カラム名 = 初期値” で宣言することで設定することができる。また、この Realm は KVC と呼ばれるキー値コーディングに対応しており、クラス中の primaryKey() メソッドでプライマリキーを設定することができる。プライマリキーを設定することで、オブジェクトを効率的に検索・更新することができ、その一意性を保つこともできる。

データの保存

```
func addRecord(record: Object) {
    let realm = try! Realm(path: getRealmPath())
    try realm.write {
        realm.add(record, update:true)
    }
}
```

データの保存には、write トランザクションの中で add メソッドを使用する。add メソッドの第 1 引数は対応する Realm オブジェクトで、その内容がレコードとして保存される。第 2 引数は保存するレコードのプライマリキーが既にデータベース内に存在していた場合に上書きするか否かを選択する要素である。true を選択した場合同じプライマリキーのレコードを上書きし、false を選択した場合は新たにレコードを作って保存する。

クエリの発行

```
1. func getCard(id: Int) -> CardData {
2.     let realm = try! Realm(path: getRealmPath())
3.     let card = realm.objects(CardData).filter("ID = %@", id)[0]
4.     return card
5. }
```

これは CardData テーブルから対応する ID のレコードを検索・取得するメソッドである。クエリの発行は 3 行目のようにして行う。まず、どのテーブルを検索するかを指定し、そのテーブル (オブジェクト) に対して filter メソッドで検索を行う。このメソッドの第 1 引数はクエリ本体で、変数を用いる場合にはその部分に %@ と記述し、第 2 引数以降に対応する値や変数を記述する。

(※文責: 細川椋太)

7.2 カードリスト

まず Storyboard で TableView と TableViewCell を使用して概形を作成した。そして、その Cell の中に写真を読み込むための UIImageView とタイトルと紹介文を表示するための 2 つの UILabel、他の画面に遷移したりお気に入りの切り替えをするためのボタンを 4 つ配置した。Cell 一つ一つが持つ情報の Model を cardData.swift という名のファイルで作成し、そこにタイトルや紹介文、画像の URL などを格納するようにした。次に setCardList という UITableViewCell のサブクラスを作成し、そこに一つのセル (以降、カードとする) に写真やテキスト、お気に入り機能の On/Off などを読み込ませるようにした。各ボタンを押したときの動作や iPhone の画面サイズを判別し、テキストのフォントサイズを変えるコーディングもこのファイル内で実装した。最後に cardList というクラスで、setCardlist で作成したカードを for 文で読み込み上から順番に並べて表示が出来るようにした。以下、カードリストを表示する上で主となるソースコードを記す。ソースコードに関しては、BigSea が掲載したサイト [7] のソースコードを参考にした。

setCardList

```
1. @IBOutlet weak var iconImage: UIImageView!
   @IBOutlet weak var title: UILabel!
   @IBOutlet weak var introText: UILabel!
2. func setCell(card :cardData) {
3. let height = UIScreen.mainScreen().bounds.size.height
4. if height >= 667 {
   self.introText.font = UIFont.systemFontOfSize(14)
5. }else {
   self.introText.font = UIFont.systemFontOfSize(12)
   }
6.self.title.text = card.title as String
7.self.introText.text = card.introText as String
8.let myImage = PhotoController().NSSImage(card.imageUrl!)
9.self.iconImage.image = myImage
   }
```

1.iconImage を写真を表示する UIImageView、title をタイトルを表示するための UILabel、introText を紹介文を表示するための UILabel として宣言。

2. セル一つに情報を加えるメソッド。引数は cardData 型の card と宣言した。card にはタイトルや紹介文、写真データが格納されている。

3.iPhone の画面の高さを求めて、その数値を height に代入。

4.iPhone6・6s の画面サイズの時の紹介文のフォントサイズを 14 に設定

5.iPhone5・5s・5c の画面サイズの時の紹介文のフォントサイズを 12 に設定

6. タイトルのテキストを代入。

7. 紹介文のテキストを代入。

8. 宣言した myImage に写真の URL を代入。
9. 写真のデータを代入。

cardList

```
1. var cards:[cardData] = [cardData]()
2. override func viewDidLoad() {
3.     self.setupLists()
4.     self.tableView.delegate = self
5.     self.tableView.dataSource = self
6. }
7. func setupLists() {
8.     for var i = 1; i <= DB().cardListSize(); i++ {
9.         let card = DB().getCard(i)
10.        let f1 = cardData(title: card.cardText!.title,
11.                           introText: card.cardText!.text, imageUrl:
12.                           NSData(data: (DB().getCard(i).photo?.photoData)!),
13.                           id: i-1, flag:DB().getFlagStatement(i))
14.        cards.append(f1)
15.    }
16. }
17. func tableView(tableView: UITableView,
18.                 cellForRowAtIndexPath indexPath: NSIndexPath) -> UITableViewCell {
19.     let cell: setCardList = tableView.dequeueReusableCellWithIdentifier
20.         ("cell", forIndexPath: indexPath) as! setCardList
21.     cell.setCell(cards[indexPath.row])
22.     return cell
23. }
24. }
```

1. cardData の配列のインスタンスを作成。
2. 画面を起動した時の動作を宣言したメソッド。
3. setupLists メソッドを実行。
4. tableView 自身をデリゲートする。
5. データの情報源を自身に代入する。
6. リストを複数作り出すメソッド。
7. カードリストの大きさ分 for 文を回す。
8. i 番目のカードの ID を受け取って宣言した card に代入。
9. 宣言した f1 にカードに必要なタイトル、紹介文、写真、お気に入りの状態やボタンなどを代入。
10. カードに代入。
11. 各行のセルを表示するメソッド。
12. cell の ID で UITableViewCell のインスタンスを作成。
13. indexPath.row 番目のカード情報を代入。
14. cell を返す。

(※文責: 山川拓也)

7.3 マップ・詳細画面

マップ・詳細画面では、マップを表示するために MKMapView、現在地を取得し表示するために CLLocationManager、マップ上に設置するピンを管理・表示するために MKAnnotationView を使用した。また、マップ・詳細画面下にあるカード情報の切り替えは UIView がもつ animateWithDuration メソッドを使用した。以下にそれぞれの核となるコードをピックアップして記載した。

MKMapView

```

1. var region:MKCoordinateRegion = self.CardMap.region
2. let location:CLLocationCoordinate2D = CLLocationCoordinate2DMake
   (DB().getCard(pic_id!).position_x, DB().getCard(pic_id!).position_y)

3. region.center = location
4. region.span.longitudeDelta = 0.005
5. region.span.latitudeDelta = 0.005
6. self.CardMap.setRegion(region, animated: true)

7. CardMap.addAnnotations(PinArray)
    
```

1. マップ起動時に表示する場所の情報を格納する変数 region の宣言。
2. マップに設定する緯度経度を格納する変数 location の初期化。ここでは、マップ・詳細画面への遷移時にタップしたカード番号を pic_id として取得し、対応するカードの緯度経度をデータベースから取得して設定している。
3. マップの中心を 2 で設定した緯度経度にする。
4. 表示する領域の横方向の縮尺を 0.005 に設定。
5. 表示する領域の縦方向の縮尺を 0.005 に設定。
6. MKMapView として宣言した CardMap に設定した中心の場所及び表示領域をセットする。なお、storyboard に MapView を設置し、コードと Outlet 接続を行ったため self.view.addSubview(CardMap) のような記述は必要がない。
7. マップ上にピンを追加する記述。引数は MKAnnotation 型であり、この場合は複数の MKAnnotation を引数としているため addAnnotations となっている。単体の MKAnnotation を追加する場合は、addAnnotation を使う。

CLLocationManager

```

1. var myLocationManager: CLLocationManager!

2. myLocationManager = CLLocationManager()
3. myLocationManager.delegate = self
4. myLocationManager.distanceFilter = kCLLocationHeadingFilterNone
5. myLocationManager.desiredAccuracy = kCLLocationAccuracyHundredMeters
6. myLocationManager.startUpdatingLocation()

7. func locationManager(manager: CLLocationManager,
   didUpdateLocations locations: [CLLocation]){
   CardMap.showsUserLocation = true
}
    
```

“Swift” Application Development Based on Field Research

1. CLLocationManager オブジェクトを格納するための変数 myLocationManager を宣言。
2. CLLocationManager をインスタンス化。
3. 位置情報を取得した時の通知先を自分自身に設定。
4. 位置情報を更新する頻度を設定。この場合は kCLLocationFilterNone を指定し、動くたびに位置情報を更新するようにしている。
5. 測位の精度を設定している。ここでは、kCLLocationAccuracyHundredMeters を指定し 100m の精度にしている。
6. GPS の使用を開始するメソッドを呼び出している。
7. 位置情報が更新された際に呼ばれるデリゲートメソッド。この中に記述してある CardMap.showsUserLocation = true は、MapView(CardMap) にユーザの位置を表示することを意味している。

MKAnnotationView

```
1. class Pin : MKPointAnnotation{
    var ID = 0
    var text = ""
    var info = ""
    var photo = NSData()
    var categoryID = 0
    var x: Double = 0.0
    var y: Double = 0.0
    var imageName = ""
}

2. for(var i = 0; i < flagTrueIDList.count; i++){
    PinArray.append(Pin())
    PinArray[i].ID = db.getCard(flagTrueIDList[i]).ID
    PinArray[i].title = db.getCard(flagTrueIDList[i]).spotName
    PinArray[i].text = (db.getCard(flagTrueIDList[i]).cardText?.text)!
    PinArray[i].info = db.getCard(flagTrueIDList[i]).info
    PinArray[i].photo = (db.getCard(flagTrueIDList[i]).photo?.photoData)!
    PinArray[i].categoryID = db.getCard(flagTrueIDList[i]).categoryID
    PinArray[i].x = db.getCard(flagTrueIDList[i]).position_x
    PinArray[i].y = db.getCard(flagTrueIDList[i]).position_y
    PinArray[i].coordinate = CLLocationCoordinate2DMake
        (PinArray[i].x, PinArray[i].y)
    PinArray[i].imageName = "fa3.png"
}

3. PinArray.append(Pin())
    PinArray[PinArray.count-1].ID = pic_id!
    PinArray[PinArray.count-1].title = DB().getCard(pic_id!).spotName
    PinArray[PinArray.count-1].text = (DB().getCard(pic_id!).
        cardText?.text)!
    PinArray[PinArray.count-1].info = DB().getCard(pic_id!).info
    PinArray[PinArray.count-1].x = DB().getCard(pic_id!).position_x
    PinArray[PinArray.count-1].y = DB().getCard(pic_id!).position_y
    PinArray[PinArray.count-1].coordinate = CLLocationCoordinate2DMake
        (PinArray[PinArray.count-1].x, PinArray[PinArray.count-1].y)

4. if((flagTrueIDList.indexOf(pic_id!)) != nil){
    PinArray[flagTrueIDList.indexOf(pic_id!)].imageName = ""
    PinArray[PinArray.count-1].imageName = "redpin.png"
}
else{
```

```

        PinArray[PinArray.count-1].imageName = "redpin.png"
    }

5. func mapView(mapView: MKMapView, didSelectAnnotationView view:
    MKAnnotationView) {
    self.UserAnnotationTap = 1
    self.CardLabelShowText = 0
    for(PinAddress = 0; PinAddress < PinArray.count; PinAddress++){
        if(PinArray[PinAddress].hash == view.annotation!.hash){
            CardSyousai.text = PinArray[PinAddress].title! + "\n" +
            PinArray[PinAddress].info
            camera2View.image = PhotoController().NSSImage
            ((DB()).getCard(PinArray[PinAddress].ID).photo?.photoData!)
            CardPoemu.text = PinArray[PinAddress].text
            break;
        }
    }
}

6. func mapView(mapView: MKMapView, viewForAnnotation annotation:
    MKAnnotation) -> MKAnnotationView? {
    let myIdentifier = "myPin"
    var myAnnotation: MKAnnotationView!
    if myAnnotation == nil {
        myAnnotation = MKAnnotationView(annotation: annotation,
        reuseIdentifier: myIdentifier)
    }
    myAnnotation.canShowCallout = true
    let cpa = annotation as! Pin
    myAnnotation.image = UIImage(named:cpa.imageName)
    myAnnotation.annotation = annotation
    return myAnnotation
}

```

1. マップ上に表示するピンをクラスとして管理するために Pin クラスを MKPointAnnotation を継承して定義した。
2. カードリスト画面でユーザが星をつけたスポット (カード) を Pin クラスを配列にした PinArray へ格納していくためのループ処理。
3. マップ・詳細画面へ遷移する際にタップしたスポット (カード) を PinArray の最後に追加する処理。
4. マップ・詳細画面へ遷移する際にタップしたスポットと星をつけたスポットが同じ場合には、星マークではなく赤色のピンを優先して表示するための処理。
5. マップ上のピンがタップされた時に呼ばれるデリゲートメソッド。PinArray からタップしたピンに対応するピンを探し出して、表示する情報を新たに設定している。
6. マップにピンを表示するためのデリゲートメソッド。このメソッドは無駄なメモリ消費を避けるために、ピンがマップ上に表示されていたら新たに作成せずにピンを再利用する設計になっている。

(※文責: 岩見建汰)

7.4 カメラ

カメラ機能の実装の方法として“AVFoundation Framework”か“UIImagePickerController”を使う2パターンあった。最初に、“AVFoundation Framework”という Swift の Framework を使用して実装しようとした。この Framework は開発者がカメラの画素数やピントの調整方法などを全て手動で制御しなくてはならなかった。特にピントの調整においては、撮影の対象物と iPhone のカメラの距離から開発者側で計算式を考えてピントが合うようにしなければならないので実装することができなかった。次に“AVFoundation Framework”を使わず“UIImagePickerController”というクラスを用いて実装した。このクラスを用いるとピントや明るさの調整はアプリ側が自動で処理をしてくれるため簡単にカメラで写真を撮る機能を実装することができた。そして、撮影した写真はデータベースに書き込むことのできない UIImage であったため、UIImage データを NSData に変換してアプリ内のデータベースに保存することにした。これにより、撮影した写真を他のクラスから参照できるようになった。以下にそれぞれ核となるコードをピックアップして記述した。

PhotoController

```

1. func ImageNSS(image:UIImage) -> NSData? {
    let data:NSData = UIImageJPEGRepresentation(image,0.8)!
    return data
}

2. func NSSImage(data:NSData) -> UIImage?{
    let decodeSuccess = data
    let img = UIImage(data: decodeSuccess)
    return img
}

3. func camerastart(){
    let sourceType:UIImagePickerControllerSourceType
        = UIImagePickerControllerSourceType.Camera
    if UIImagePickerController.isSourceTypeAvailable
        (UIImagePickerControllerSourceType.Camera){
        let cameraPicker = UIImagePickerController()
        cameraPicker.sourceType = sourceType
        cameraPicker.delegate = self
        self.presentViewController(cameraPicker,
            animated: true, completion: nil)
    }
}

4. func imagePickerController(imagePicker: UIImagePickerController,
    didFinishPickingMediaWithInfo info: [String : AnyObject])
    if let pickedImage = info[UIImagePickerControllerOriginalImage]
    as? UIImage {
        let image:UIImage! = pickedImage
        let appDelegate:AppDelegate = UIApplication.
            sharedApplication().delegate
            as! AppDelegate
        let pic_id = appDelegate.P_ID
        if(image != nil){
            DB().addPhoto(pic_id!, photoData: ImageNSS(image!))
            let photoID = DB().getLastPhotoID()
            DB().linkToCard(photoID)
        }
    }
}

```

```

        imagePicker.dismissViewControllerAnimated(true, completion: nil)
    }

```

1. UIImage 型の撮った画像を NSData 型に変換するメソッド。データベースで扱える型の中に NSData 型があったため保存する型として選択した。引数として UIImage 型の撮った画像データを受け取り、80 %圧縮をかけて NSData 型に変換して返す。
2. NSData 型のデータを UIImage 型のデータに変換するメソッド。データベースに保存された画像データは NSData 型なので画像として表示するためには UIImage 型に変換する必要がある。
3. UIImagePickerController を呼び出すメソッド。まず、iPhone のカメラが利用可能か調べる。もし、利用可能なら現在の画面からカメラの撮影画面に遷移する。
4. UIImagePickerController は Apple が用意しているメソッドで、写真撮影ボタンタップ撮影が完了するときに呼び出される。まず、カードリストで選択した ID を取得する。データベース上から、その取得した ID に対応した画像データ項目に撮った画像データを書き込む。以上の処理が終わると UIImagePickerController を閉じる。

(※文責: 池田俊輝)

7.5 振り返る機能

振り返る機能は、デフォルトの写真とユーザが撮影した写真の 2 種類を見ることができるよう UIViewController を 2 つ用意し、アニメーションなしの画面遷移を行うことで 1 つの画面で切り替えているように見せている。それぞれの UIViewController にはたくさんの画像などを並べて表示する際によく使用される UICollectionView 及び 2 つの選択を排他的に行うパーツである UISegmentedControl を設置し、xib ファイルを使用してセルを設定した。以下に振り返る機能で使用した主な技術である UICollectionView、UISegmentedControl,xib についての詳細を記載した。

UICollectionView

```

1. func collectionView(collectionView: UICollectionView,
    numberOfItemsInSection section: Int) -> Int {
    if(appDelegate.pic_segmented == 0){ // 撮影写真を選択している場合
        if(DB().getUpdatedCardIDArray().count == 0){
            return 1
        }else{
            return DB().getUpdatedCardIDArray().count // 枚数分の表示
        }
    }else{ // デフォルト写真を選択している場合
        return DB().cardListSize()
    }
}

2. func collectionView(collectionView: UICollectionView,
    cellForItemAtIndexPath indexPath: NSIndexPath) -> UICollectionViewCell {
    if(appDelegate.pic_segmented == 0){ 撮影写真の場合
        let cell = collectionView.dequeueReusableCellWithReuseIdentifier
            ("PagingCollectionViewCell", forIndexPath: indexPath) as!
            PagingCollectionViewCell
    }
}

```

```
if(DB().getUpdatedCardIDArray().count == 0){
    collectionView.scrollEnabled = false
    let kikoimage = UIImage(named: "kiko.png")
    cell.photo.image = kikoimage
    cell.titleLabel.text = ""
    cell.introLabel.text = ""
    cell.NoPhotoLabel.text = 撮影写真がありません。＼n 写真を撮ると表示されます。

    credit.frame = CGRectMake(70, 310, 300, 120)
    credit.font = UIFont(name: "HiraginoSans-W3", size: 10.0)
    self.addSubview(credit)
    credit.text = " 木古内町観光マスコットキャラクターキーコ "

    return cell
}else{
    var UpdateCardIDArray1 = DB().getUpdatedCardIDArray()
    let NSphotodata1 = DB().getCard(UpdateCardIDArray1
[indexPath.row]).photo?.photoData
    cell.photo.image = PhotoController().NSSImage(NSphotodata1!)

    let titletext1 = DB().getCard(UpdateCardIDArray1
[indexPath.row]).cardText?.title
    let introtext1 = (DB().getCard(UpdateCardIDArray1
[indexPath.row]).cardText?.text)!

    cell.titleLabel.text = titletext1
    cell.introLabel.text = introtext1
    cell.NoPhotoLabel.text = ""

    return cell
}

}
}else{ // デフォルト写真の場合
    self.collectionView.scrollEnabled = true
    let cell = collectionView.dequeueReusableCellWithReuseIdentifier
("PagingCollectionViewCell", forIndexPath: indexPath) as!
PagingCollectionViewCell
    let cardcounts = DB().cardListSize()
    var IDArray2:[Int] = []

    for(var i = 1; i <= cardcounts; i++){
        IDArray2.append(i)
    }

    let NSphotodata2 = DB().getDefaultPhoto
(IDArray2[indexPath.row]).photoData

    // 写真データ(NSdataを)に変換image
    cell.photo.image = PhotoController().NSSImage(NSphotodata2!)

    let titletext2 = DB().getCard(IDArray2[indexPath.row]).
cardText?.title
    let introtext2 = (DB().getCard(IDArray2[indexPath.row]).
cardText?.text)!

    cell.titleLabel.text = titletext2
    cell.introLabel.text = introtext2
    cell.NoPhotoLabel.text = ""

    return cell
}
}
```

```
}

```

1. UICollectionView に表示するセルの数を返すメソッド。撮影写真が 0 枚の場合は、木古内町観光マスコットキャラクターキーコを表示するために 1 を返す。0 枚以外の場合は、撮影した枚数を返す。
2. UICollectionView のセルに表示する画像とテキストを設定するメソッド。撮影写真を表示する UIViewController になっている且つ撮影写真が 0 枚の場合は木古内町観光マスコットキャラクターキーコを設置したセルを返す。撮影写真を表示する UIViewController になっている且つ撮影写真が 1 枚以上ある場合は、ユーザが撮影した写真を設置したセルを返す。デフォルトの写真を表示する UIViewController になっている場合は、デフォルトの写真を設置したセルを返す。

UISegmentedControl

```
1. @IBAction func segmentedchanged(sender: AnyObject) {
    switch sender.selectedSegmentIndex{
        case 0: // 撮影写真
            appDelegate.pic_segmented = 0
            self.dismissViewControllerAnimated(false, completion: nil)

        case 1: // デフォルト写真
            appDelegate.pic_segmented = 1

        default:
            break
    }
}

2. @IBAction func segmentedchanged(sender: AnyObject) {
    switch sender.selectedSegmentIndex{
        case 0: // 撮影写真
            appDelegate.pic_segmented = 0

        case 1: // サンプル写真
            appDelegate.pic_segmented = 1
            let targetViewController = self.storyboard!.
            instantiateViewControllerWithIdentifier("album_defaultphoto")
            self.presentViewController( targetViewController ,
            animated: false, completion: nil)
            self.segmented.selectedSegmentIndex = 0 // 選択を戻す

        default:
            break
    }
}

```

1. こちらの IBAction はデフォルト写真を表示する UIViewController に設置されている Segmented に接続されている。撮影写真を選択した場合は該当する UIViewController に画面遷移を行うが、デフォルト写真が選択された場合は何もしない。
2. こちらの IBAction は撮影写真を表示する UIViewController に設置されている UISegmentedControl に接続されている。1 の IBAction と逆の動作を行う。

xib

```

1. class PagingCollectionViewCell: UICollectionViewCell {

    @IBOutlet weak var photo: UIImageView!
    @IBOutlet weak var titleLabel: UILabel!
    @IBOutlet weak var introLabel: UILabel!
    @IBOutlet weak var NoPhotoLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        NoPhotoLabel.numberOfLines = 2
        titleLabel.numberOfLines = 1
        introLabel.numberOfLines = 30
    }
}

2. self.collectionView.registerNib(UINib(nibName:
"PagingCollectionViewCell", bundle: nil), forCellWithReuseIdentifier:
"PagingCollectionViewCell")

self.addSubview(self.collectionView)

```

※ xib は Story board がアプリ全体の画面遷移を管理するのに対し、1つ又は複数の画面についての設計を行い Story board へ組み込むような形で使用する。xib を画面ごとに分割してそれぞれの xib を Story board に組み込むような挙動を設定することでチーム開発でのコンフリクトが発生しにくくなり、効率良く開発ができる。

1. xib に使用している部品とソースコード間での IBOutlet 接続及び UILabel 内に表示するテキストの行数を設定している。
2. xib を読み込むコードである。xib のファイル名を nibName に String で指定し Identifier も他と重複しない名前を String で指定する。読み込んだ後は通常の Story board と同様に addSubview を使用して画面へ追加・表示する。

(※文責: 岩見建汰)

7.6 印刷

印刷するリーフレットの画像データを iPhone の機能である AirPrint を使って無線で送信し、リーフレットを印刷する。AirPrint を使用するためには画像データを NSData にする必要があったためリーフレットの画像を pdf に変換し、その pdf データを NSData に変換することにした。両面印刷するためには2つの画像データを1つのものとして扱う必要があったため、pdf にするプロセスを挟んだ。しかし AirPrint に関して、Objective-C で実装するための文献は数多く存在するものの、Swift での実装例は少なかった。そのため、Objective-C で書かれたコードを1行ずつ Swift の文法で書き直していくことでこの機能を実装した。印刷ボタンをタップした後は、AirPrint を行うための設定画面が表示される。そこではプリンタを選択、印刷枚数、白黒印刷にするのかカラー印刷にするのか選択することができる。ユーザが AirPrint の設定をし、印刷ボタンを選択すると Wi-Fi でデータがプリンタに送信され印刷が開始される。以下にそれぞれ核となるコードをピックアップして記述した。

画像データを PDF にする処理

```
1.let path = arrayPaths[0] as NSString
2.let fullname : NSString = (filename as String) + ".pdf"
3.let pdfFilename = (path as String) + "/" + (fullname as String)
4.UIGraphicsBeginPDFContextToFile(pdfFilename, CGRectZero, nil)
5.UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 1690, 1195), nil)
6.let point1 = CGPointMake(0, 0)
7.front.image?.drawAtPoint(point1)
8.UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 1754, 1240), nil)
9.let point2 = CGPointMake(0, 0)
10.back.image?.drawAtPoint(point2)
11.UIGraphicsEndPDFContext()
12.myData = NSData(contentsOfFile: pdfFilename)!
```

1. 保存するディレクトリのパスを宣言する
2. pdf ファイル名を宣言する
3. パス名を含めた pdf ファイル名の宣言する
4. pdf を作成する詳細な設定を決めるために宣言する
5. 1 枚目の pdf ファイルはページの大きさを指定する
6. 1 枚目の pdf ファイルを書き込む座標の開始時点を指定する
7. 1 枚目で指定した座標を設定に反映させる
8. 2 枚目の pdf ファイルはページの大きさを指定する
9. 2 枚目の pdf ファイルを書き込む座標の開始時点を指定する
10. 2 枚目で指定した座標を設定に反映させる
11. pdf の処理を終える
12. pdf を NSData に変換する

印刷する処理

```
@IBAction func print(sender: AnyObject) {
    1. if(UIPrintInteractionController.canPrintData(myData)){
        2. let printController = UIPrintInteractionController.
           .sharedPrintController()
        3.printInfo.outputType = UIPrintInfoOutputType.General
        4.printInfo.duplex = UIPrintInfoDuplex.LongEdge
        5.printInfo.orientation = UIPrintInfoOrientation.Landscape
        6.printController.showsPageRange = true
        7.printController.printingItem = myData
        8.printController.presentAnimated(true,
           completionHandler: nil)
    }
}
```

1. 取得したデータが AirPrint できるデータであれば実行する
2. AirPrint の設定ができる画面を宣言する
3. 印刷をカラーか白黒か選べるようにする
4. 両面印刷の長辺とじに設定する
5. 印刷する向きを横向きにする
6. 印刷する向きを反映する
7. プリントするデータを設定する
8. AirPrint の設定ができる画面に遷移する

(※文責: 池田俊輝)

第 8 章 今後の課題と展望

8.1 リリースについて

最終成果発表会で得られた意見をもとに第 4 サイクルの改善フェーズを行った後、続く第 5 サイクルでアプリケーションへ反映させる予定である。リリース時期は 2 月中を目標としている。この時期にリリースの目標を設定した理由としては、3 月に北海道新幹線が開業することが挙げられる。観光客が最も訪れると予想される開業直後に間に合うようリリース準備を進めていく予定である。今後の作業としては、まずレビューの結果をもとに機能の拡張や UI の再設計を行い、並行して既知のバグの修正を行っていく。これらが完了し次第、App Store へリリースする。また、リリース後にも定期的なメンテナンスを行っていくほか、季節ごとのコンテンツの追加などの案も検討されている。なお、メンテナンスの方法や継続期間についての検討も開発と並行して行っていく予定である。



図 8.1 今後の予定

(※文責: 横山翔栄)

8.2 ステークホルダーへの報告会について

我々はキーコ紀行をリリースするにあたって木古内町長と木古内商工会「木古内町観光協会」へ最終成果物について報告すると共にキーコ紀行の運用についても相談を行う予定である。以下に現段階で考案している報告内容と会議内容を示す。

報告内容

- 作成背景
- 観光客と木古内町のメリット・デメリット
- キーコ紀行の使い方を説明 (プリンタ持参)
- キーコ紀行のユーザストーリー

協議内容

- キーコ紀行内に記載している店舗情報を我々がお店から直接収集することについて問題がないか

“Swift” Application Development Based on Field Research

- キーコ紀行で使用している写真の使用許可について
- 「木古内観光交流センターみそぎの郷きこない」にプリンタを設置して運用することに問題がないか
- キーコ紀行内の観光情報を逐一アップデートし、継続運用することについて
- 観光スポット紹介文の是非について
- その他、キーコ紀行全般の是非について

(※文責: 岩見建汰)

第9章 学び

9.1 グループ間の情報共有

前期から講義の最後の20分は各グループで進捗報告を行うようにしていた。報告の内容は、その日に行った活動とその活動がどういった進捗を生んだのかを報告し合うものであった。前期は、どのグループも基本は要件定義の段階で止まっていたため、質問があるかと聞かれてもいつまでに誰がどのようなことを行うのか程度の受け答えしか行っていなかった。まずここに第一の反省点があった。今どのようなことが問題となっていて、どう解決していいのかわからなくて困っているかを相談することが無かったということである。このような相談をグループ間で行っておけば、何か別の解決策が生まれて早期段階で問題解決に近づけたのかもしれない。

後期、第3サイクルの時点では我々観光グループと医療グループは実装に取り掛かり始めていた。しかし、進捗報告の時間では出来たものを見せ合うことはしておらず、依然としてただプロジェクトの活動中に行ったことを報告し合うだけでいた。ここで第二の反省点が見られた。担当教員に指摘されるまでアプリを他のグループのメンバに見せ合うことがなかったため、同じ技術をそれぞれのグループが利用していて、片方が習得しているのにもう片方が詰まっていた、それに気づくことに遅れてしまったのである。具体的には両グループ共に用いていたカメラ機能に関してである。こちらで実装していたカメラの機能の使用画面の言語が英語になってしまい、その問題の解決策が見出せないままだった。休憩時間にメンバが医療グループにアプリを見せてもらいに行ったとき、医療グループでも我々と同じカメラ機能を実装しており、さらにこちらの抱えていた問題を解決済みであった。それにより、こちらが抱えていた問題を解決することができた。

第4サイクルに入ってから、前サイクルで担当教員に「アプリの画面ができていながら、できているところまででもプロジェクターとかで映してデモを見せたほうが良い。」とアドバイスを頂いたので、それを実行した。これにより、TAや教員だけでなく、学生間でのアドバイスを行えるようになった。それは技術面だけに限らず、ボタン・画像の配置やアプリ内のデザインの配色などのUI面でもお互いに見直せる機会が増えた。特に教育グループはアニメーションを作っていたので、そのデモを見ながら一つ一つのシーンについて全員で吟味して欠陥をプロジェクトメンバ全員で指摘することができた。

プロジェクトの活動は終わってしまったが、情報共有をしておくことは今後も活かせる学びであると言えるだろう。研究室ではメンバや教員と、就職先では本物のプロジェクトとして活動をしていくことになる。その時、今回の反省点である相談をすることと報告・連絡をすることが重要になってくるだろう。

(※文責: 山川拓也)

9.2 計画管理の必要性

プロジェクト学習を通して計画することとそれを継続的に管理することの重要性を学んだ。計画の管理をうまく行えていなかったことによる最も大きな失敗は、第4サイクルでキーコ紀行の実装を行ったときにあった。第4サイクルの開発開始時にタスクの洗い出しを行い、そのタスクの管理

のために WBS（ワークブレイクダウンストラクチャ）を作り運用のルールなども決めたが、それを継続的に運用するには至らなかった。そのため、どの機能を実装するのか、その優先度はどの程度か、それを誰がやるのか、といった計画の管理が上手くいかず、それぞれのメンバに認識のずれが生じてしまったことがあった。また、メンバそれぞれが現在持っているタスクの把握もできていなかった。その認識のずれの解消やタスクの把握をするために本来必要でなかったはずのコミュニケーションに時間を割いてしまった経験から、継続的に計画管理を行うことの必要性に気付いた。理想的には、WBS やタスクかんぱんなどの形式に則ったタスク管理を、チームのメンバ全員が行うべきだと考えるが、継続的に運用するためには、その重要性を周知し、運用を習慣化させるとともに、より手間をかけずに管理できるような工夫が必要である。しかし、具体的に運用方法をどうすべきかという点に関しては未だ不明瞭なままであるため、その枠組みを考えることが今後の課題である。

(※文責: 細川椋太)

9.3 バージョン管理

プロジェクト発足時に GitHub を用いたソースコードのバージョン管理についての勉強会に参加した。勉強会では Git コマンドの使用を実践的に学び、第 1 サイクルでは GitHub を使用してバージョン管理を行うことができた。また、Git で使用するコマンドをマニュアルとして作成(付録 C.2) しチームで共有した。

第 2 サイクルは中間発表まで 1 週間を切っていた、コーディング担当者である 2 人は日常的に会う頻度が高いメンバ同士だった、GitHub に対する知識が浅く第 1 サイクル時にマージした際にファイルが消えるなどのトラブルへの対処法が分からないままだったという要因が重なり、Google Drive^{*1} や Mac に搭載されている AirDrop^{*2} を使用してファイル共有を行いコーディングを進めていた。その際は入念にマージや分担について話しあったので大きなトラブルは発生しなかったが、時間を大幅に消費してしまった。その消費してしまった時間をコーディングや成果物の改善に関する議論に費やすことができたなら中間成果発表時での成果物の質を高くすることができたと反省している。そして夏季休業終了後の第 3 サイクルへと突入したが、コーディングを行っていないため GitHub は使用していない。

第 4 サイクルは GitHub を使用してコーディングを進めることができたが、GitHub を徐々に使用するということもあり導入に戸惑ってしまった。原因として GitHub に関する技術・知識の量がメンバ間で異なっていたことや、その事実を早めに解決すべきだったことがあげられる。GitHub を使用してコーディングを進めることで誰がいつどこを変更しているかだけでなく、タスクのおおまかな進捗状況を直接会うことなく把握できる利点に気がついた。

初期段階から GitHub を使用していればファイルのバージョン管理における混乱を避けられたかもしれない。仮に第 4 サイクルから GitHub を使用することになっていたとしても夏季休業中や第 3 サイクルの時点で GitHub を何かしらの形で使用していればスムーズに第 4 サイクルで GitHub を活用して開発を進めることができたと考えられる。

(※文責: 岩見建汰)

^{*1} Google のオンラインストレージ。無料で 15GB 使用することが可能である。

^{*2} AirDrop は Wi-Fi および Bluetooth を使用して、ファイルの共有を簡単にできるサービスである。AirDrop は Apple Inc. の商標である。

9.4 ポスター

本プロジェクトを通して、成果発表のために計3回のポスター製作を行った。これらのポスターについてもアプリと同様にレビューを受け、改善を図っていった。その中で、空白や図表の必要性などグラフィックデザインの要素について学ぶことができた。まず、空白とは何もない無のスペースではなく、複数の要素のグループ化や構造化などの役割を持つ要素の一つであることが分かった。これはゲシュタルト心理学におけるプレグナンツの法則のうち近接の要因と関係し、この法則については昨年度の認知心理学の講義で学習したものであるが、プロジェクトの中で行ったポスター製作によって実践的な学びを得ることができた。また、ポスターを図表を中心に組み立て、文章の分量を抑えることで、誘目性が高く魅力的なポスターになることが分かった。文章の分量が多いポスターは閲覧者に読む意欲を失わせる。これはプロジェクトの成果を伝え、意見や感想をもらう機会を失うことにつながる。一方で、文章の分量を極力抑え図表を多く使用したポスターは、閲覧者に重い印象を与えず、気軽に読んでみようという意欲を起こさせる。とりわけポスターセッションにおいては図表を用いた口頭発表を行うため、冗長な文章は不要である。簡潔な文章と図によって、ポスターは読まれやすく、伝わりやすいものになることを学んだ。

(※文責: 横山翔栄)

第 10 章 到達目標に対する評価

10.1 機能の評価

アプリケーションに実装した機能は、観光情報へのアクセス性を高め、撮影した写真を用いて観光の体験の振り返りを促すことを目標としていた。観光情報へのアクセス性を向上させるために実装した「観光する」機能は、発表会などで得られたレビューにより、既存の Web ページや SNS と比較し情報を得やすいとの評価を得た。以下に、最終発表会で行ったアンケートの「このアプリの良いと感じた点は何ですか？」の項目から、観光情報へのアクセス性に関するコメントを抜粋した。

- 木古内で出来ることについて、グルメや観光スポットについて幅広く見られる。
- 観光地の内容がわかりやすく伝わること。

これらのコメントから、観光情報へのアクセス性の向上についてはおおむね到達目標に達したものと評価できる。しかしながら、「今自分がいる場所から近い場所を探したい」「検索機能が欲しい」などの課題の指摘も受けた。今後、リリースへ向けた開発にあたっては、これらのレビューを踏まえて開発を行っていく。以下に、同アンケートの「このアプリの悪いと感じた点は何ですか？」の項目から観光情報へのアクセス性に関するコメントを抜粋した。

- 検索機能がないので、分野別に見たい時は使いにくい。
- アクセスの仕方も見られると良かった。

撮影した写真を用いて観光の体験の振り返りを促すために実装した「印刷する」機能は、自分で撮影した写真でリーフレットを作成できる点が「新しい」「思い出を形に残せる」との評価を得た。以下は同アンケートの「このアプリの良いと感じた点は何ですか？」の項目から、「印刷する」機能に関するコメントの抜粋である。

- データだけではなく物で思い出を残せる点。
- ありそうでないアプリ。着眼点は面白い。
- 自分好みのリーフレットを作れたり、「自分だけ」のものが作れるのがいいと感じた。

以上から、機能に有用性があり、ユーザの興味関心を引くことができしており、写真を用いた観光の振り返りを十分に促すという目標に到達したと評価できる。ただし、「自分でレイアウトを作りたい」「新規ユーザの獲得はどうするか」といった指摘もあり、これらの解決が今後の課題となっている。以下は同アンケートの「このアプリの悪いと感じた点は何ですか？」の項目から、「印刷する」機能に関するコメントの抜粋である。

- 思い出はできるが、新規の人が来ない。
- 元の写真に戻すときのやり方が分からない。
- 自分なりのレイアウトが作れない。

10.2 活動内容の評価

本プロジェクトでの活動にあたって、到達目標として「木古内町へのフィールドワークで得られた経験を重視したアプリケーション設計」「スクラムの実践」「Swift や Git、データベースの技術習得」「UI や UX などユーザビリティの考慮」を掲げた。まずアプリケーション設計に関しては、フィールドワークから得た「観光情報の一元化が必要」という気づきを盛り込んだものの、アプリケーションの大枠としてはフィールドワーク以外の部分から着想を得たものが多い。結果として、「木古内町ならでは」という部分が最後まで薄くなってしまったものと推察され、目標を達成したとは言い難い。また、スクラムの実践については、前期の一時期のみ形式に則ったスクラムの実践を行った以外、あいまいな形で進行していた。このことにより、スクラムの方法論については習得したものの、実践に関しては十分な経験を得られていないものと評価する。Swift などの開発技術習得については、各メンバーが担当箇所の技術について調査を行い、また実装を行うことができていたため、目標を十分に達成したと評価する。最後に UI や UX の検討について、UX に関してはユーザの視点に立った検討をプロトタイプなどを用いて多く行うことができたものと評価する。一方で、UI 設計の際の資料の調査については不十分といえる点が多く、主観的評価の下に実装を行った点が多いことは否めない。その結果、ボタンのように見える紛らわしい部分など、UI に欠陥ともいえる部分が残ってしまった。以下に、前項と同じアンケートで得られたレビューから UI に関するコメントを抜粋する。

- 押せそうだけど押せないパーツがあったり、押したときのレスポンスが悪い部分があった。
- 唐突に印刷画面になる。
- UI デザインが見やすくわかりやすい。観光地で写真を撮ってリーフレットにするのは今までにない発想で観光の思い出がより残って良いと思った。

全体として、アプリケーションの機能については概ね目標を達成しているが、プロジェクトの活動内容自体は十分に目標を達成しているとは言えない部分が多く残ったと言える。

(※文責: 横山翔栄)

第 11 章 まとめ

11.1 まとめ

本プロジェクトでは、フィールドを調査することによって問題点を発見し、その問題点を解決するためのアプリケーションを開発して、地域・社会に貢献することを目標として活動を行った。開発には、その目標を達成する質の高いアプリを開発するために評価と改善を繰り返すアジャイル開発手法を用いた。

観光系グループでは、木古内町の観光をフィールドと設定して調査を行い、その調査結果をもとに分析を行って「観光情報のアクセス性改善と魅力の発信」「観光した際に撮影した写真を振り返る機会が少ない」といった課題を定義して、それを解決するために iOS アプリ「キーコ紀行」を開発した。このアプリは、「木古内町でできること」に着目した観光情報の利用と、観光スポットで撮影した写真を用いてオリジナルリーフレットの作成ができるアプリケーションである。このアプリを開発する過程で 4 つの PDCA サイクルを経て、その活動経験とたくさんの評価やアドバイスから、グループメンバそれぞれが要件定義や実装、マネジメントなどプロジェクト活動全体に関する数多くの学びを得ることができた。

今後、この「キーコ紀行」は観光グループのメンバが引き続き開発を行い、2016 年 2 月中にリリースする予定である。このアプリをリリースをするために未実装の機能の実装を行い、その後に木古内町長と木古内商工会「木古内町観光協会」へ最終成果物について報告するとともにキーコ紀行の運用について相談を行う予定である。リリース後には実際に木古内でこの「キーコ紀行」を使っていた方からのレビューなどを受け、よりよいアプリへと改善するとともに継続的な運用も行う予定である。

(※文責: 細川椋太)

付録 A その他新規習得技術

A.1 WebAPI

アプリケーションの作成にあたり、天気の情報を取得するために WebAPI を利用した。これは HTTP などの Web 技術を用いてプログラムの提供する機能を利用するためのサービスであり、有償のものも含め多くの種類が公開されている。本プロジェクトの前期成果物には “OpenWeatherMap”^{*1} を利用しており、この API から返される JSON 形式のデータを処理することで、天気情報を表示している。WebAPI の優れている点は、インターネットに接続された環境であればどこからでも利用できるという点である。前述の通り、WebAPI は HTTP などの Web 技術を用いたサービスである。このため、特殊な機器やシステムを一切使用することなく、多くの機器で普遍的に利用することができる。

(※文責: 横山翔栄)

^{*1} <http://openweathermap.org/>

付録 B 活用した講義

報告書の参考文献を明示するために、科学技術リテラシーで学んだ参考文献の書き方の知識を用いた。また、中間発表で用いたポスターを作成するにあたって、情報表現基礎や情報デザインの講義で得た知識や経験を用いた。見やすく分かりやすいポスターを作るために、できる限り文字を図にして置き換え、図解によって主要を伝えるという点を意識した。また、全体の印象を整えるため各要素の位置を他の要素にそろえた。プロジェクトを進めていくうえでアプリの開発形態を見直すとき、ソフトウェア設計論で学んだウォーターフォールモデルとアジャイル開発手法の考え方を参考にした。本プロジェクトはアジャイル開発手法であり、ウォーターフォールモデルのような1つ1つの段階を固めてから次の段階に進むということはず、開発を迅速に行いレビューを受け、再び開発をしていくものであるということを常に意識することが重要だと学んだ。アプリはどのような機能を持っているか、ユーザはどのようにアプリを使うのかを明示してそれを設計するためにソフトウェア設計論のユースケース図とアクティビティ図の書き方を参考にした。以下が作成した図である。

(※文責: 山川拓也)

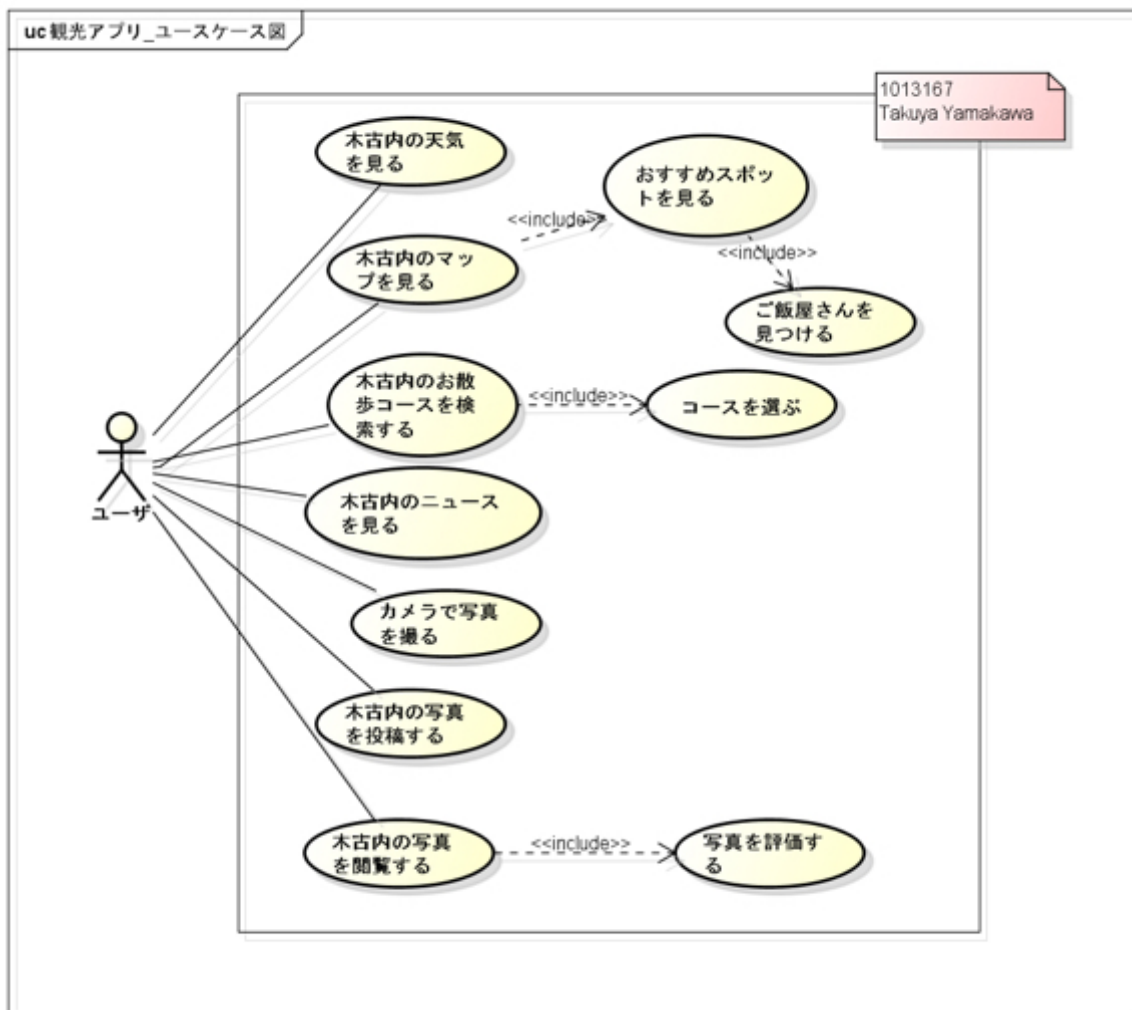
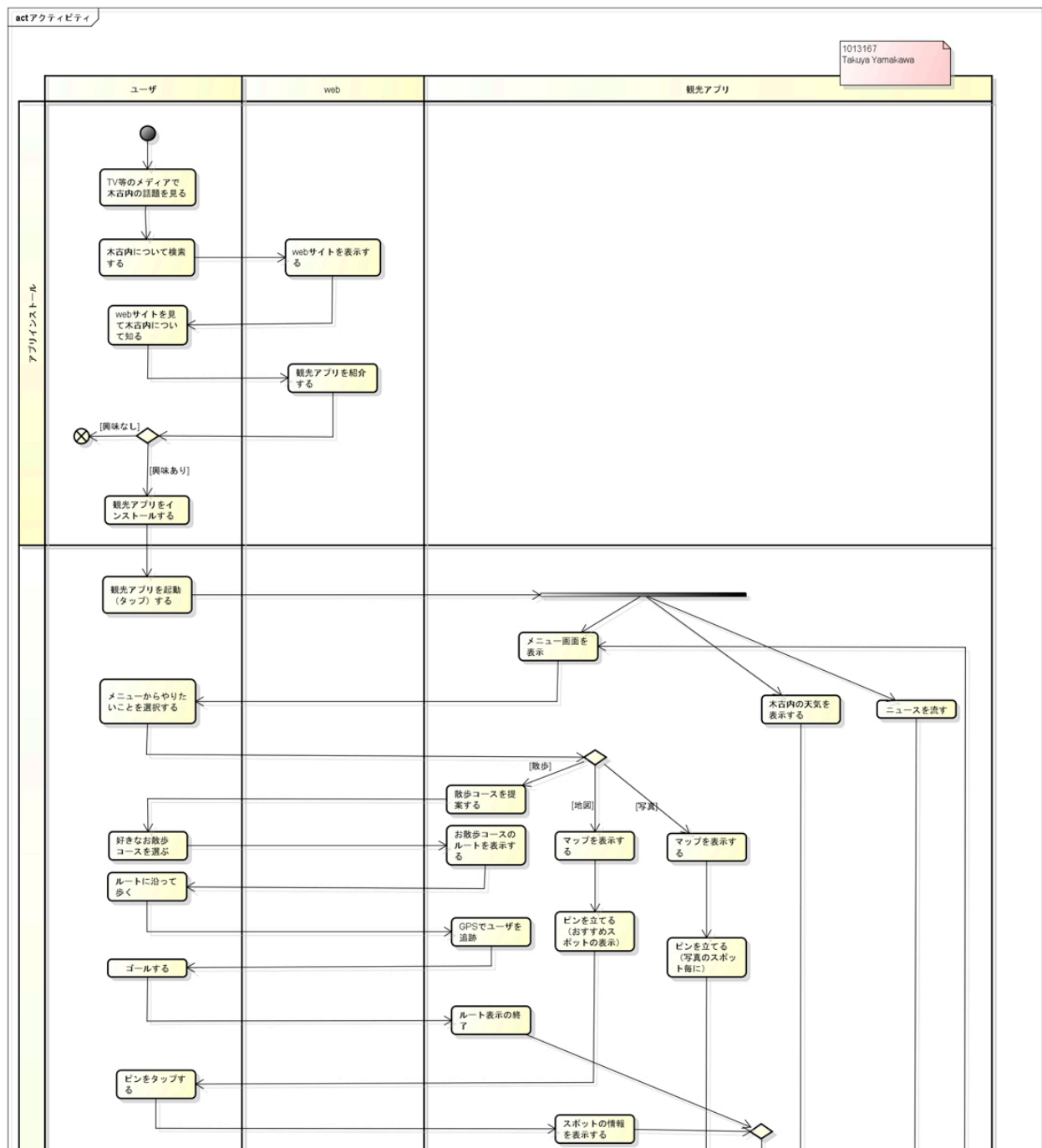


図 B.1 ユースケース図



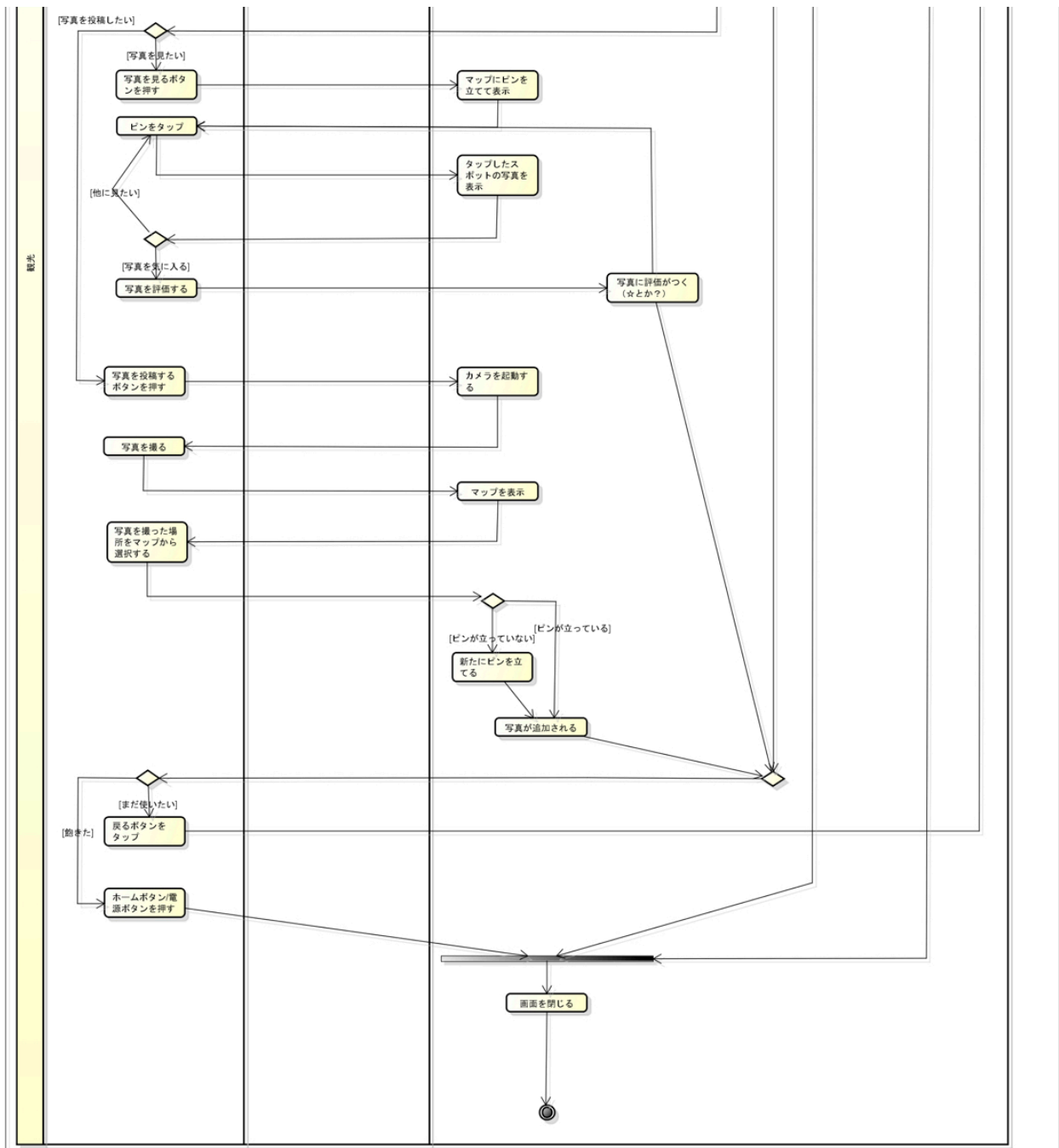


図 B.2 アクティビティ図

付録 C その他成果物

C.1 Git マニュアル

運用ルールとか

- ・プルリクエストは、コメント欄に3人の承認があって初めてマージできるものとする。マージするのは、原則として3人目のコメント者が行う。(あまりにも遅い場合は残りの2人がマージ)
- ・ブランチの切り方については「git-flow」を参照のこと

Git を使った作業の始め方

cd [場所] でどこで作業するか決めて移動する。

git clone [URL] でクローンする。(初回のみ ※1)

git branch develop ローカルに develop ブランチを作る

git checkout develop (develop ブランチに pull するため、移動)

git pull origin develop でリモートリポジトリの内容をローカルに反映

git branch で全ブランチ、現在のブランチを確認。

git branch yamakawa で作業ブランチ” yamakawa” を作成。

git checkout yamakawa でブランチの移動。

git checkout -b yamakawa

→ (yamakawa ブランチを作成して移動)

ファイルを編集・保存

git add [file]

git commit -m “commit message”

git push origin yamakawa -> yamakawa ブランチを push

※1 もし、push が 403 エラーで成功しない場合、clone するときに github ユーザ名を付けてみる

git clone https://Github ユーザ名@github.com/FUNPBL2015/kan_practice1.git

git pull origin pull したいリモートブランチ名:ローカルブランチ名

GitHub 文化の話…

プルリクエスト承認コメントの慣例「LGTM(Looks Good To Me)」

<http://d.hatena.ne.jp/keyword/LGTM>

lgtm.in とかも調べてみる

ローカルの master がリモートの master より遅れているときの解消方法

```
$ git graph
```

```
(origin/master, origin/HEAD)
```

```
—
```

```
— * (HEAD, origin/develop, develop)
```

— /

(master)

↑この状態のとき

git checkout master

git pull origin master

で、ローカルの master がリモートと同じ位置まで移動する。

リモートリポジトリが壊れた場合、ローカルリポジトリを丸ごとコピーする形で修復を行うこともある。バックアップの意味でも、↑のような状況になっている場合はこの作業を行うこと。

こんな時.....

• git status をして、

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout - <file>..." to discard changes in working directory)

modified: kankouApp.xcodeproj/project.xcworkspace/xcuserdata/

kouritsuhakodatemiraidaigakukoudoictkosu.xcuserdatad/UserInterfaceState.xcuserstate

と表示されたら、

git checkout - <赤字をコピー> すればうまくいける。

• ローカルのブランチを消したい

git branch -d [ブランチ名]

• git push origin :test

リモートにある test ブランチを削除できる。

• 今いるブランチのローカルでの変更をなかったことにする

変更をコミットしてからブランチ切り替えして、と言われた時に使える

コンフリクトの時は！

エラーメッセージに従って処理

参考文献

- [1] 木古内町. 木古内町観光協会. <http://kikonai-kankou.net/> (2015/7/22 アクセス)
- [2] gitflow cheatsheet. danielkummer.github.io.
http://danielkummer.github.io/gitflowcheatsheet/index.ja_JP.html
(2016/01/06 アクセス)
- [3] Time Out Tokyo. 札幌でしかできない 50 のこと - Time Out Tokyo (タイムアウト東京).
<http://www.timeout.jp/tokyo/ja/things-to-do/sapporo>
(2016/01/19 アクセス)
- [4] 富士フィルム株式会社. フジフィルムのフォトブックでフォトアルバムを作成
<http://f-photobook.jp/> (2016/01/19 アクセス)
- [5] 株式会社ルックバイス. 「カタログ」と「パンフレット」と「リーフレット」の違い - 違いが分かる辞典.
<http://chigai-allguide.com/%E3%82%AB%E3%82%BF%E3%83%AD%E3%82%B0%E3%81%A8%E3%83%91%E3%83%B3%E3%83%95%E3%83%AC%E3%83%83%E3%83%88%E3%81%A8%E3%83%AA%E3%83%BC%E3%83%95%E3%83%AC%E3%83%83%E3%83%88/> (2016/01/19 アクセス)
- [6] Realm. Y Combinator. 2015. <https://realm.io/jp/>. (2016/1/18 アクセス)
- [7] BigSea. Swift で CustomCell を作って画像付きリスト表示.
<http://qiita.com/BigSea/items/9aa35b95e5d4d1dc8a52>. (2016/01/07 アクセス)