公立はこだて未来大学 2021 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2021 Systems Information Science Practice

Group Report

プロジェクト名

Let's リモートセンシング！

Project Name

Let's Remort Sensing!

グループ名

グループ 1

Group Name

Group 1

プロジェクト番号/Project No.

18

プロジェクトリーダ/Project Leader

野原広大　Kohdai Nohara

グループリーダ/Group Leader

山田純平　Junpei Yamada

グループメンバ/Group Member

飯田珠羅　Jura Iida
外川雄也　Yuya Togawa
山田純平　Junpei Yamada
蓬畑尚輝　Naoki Yomogihata
菅原慎哉　Shinya Sugawara
山本陽平　Yohei Yamamoto

指導教員

長崎健　和田雅昭

Advisor

Takeshi Nagasaki　Masaaki Wada

提出日

2022 年 1 月 19 日

Date of Submission

January 19, 2022

概要

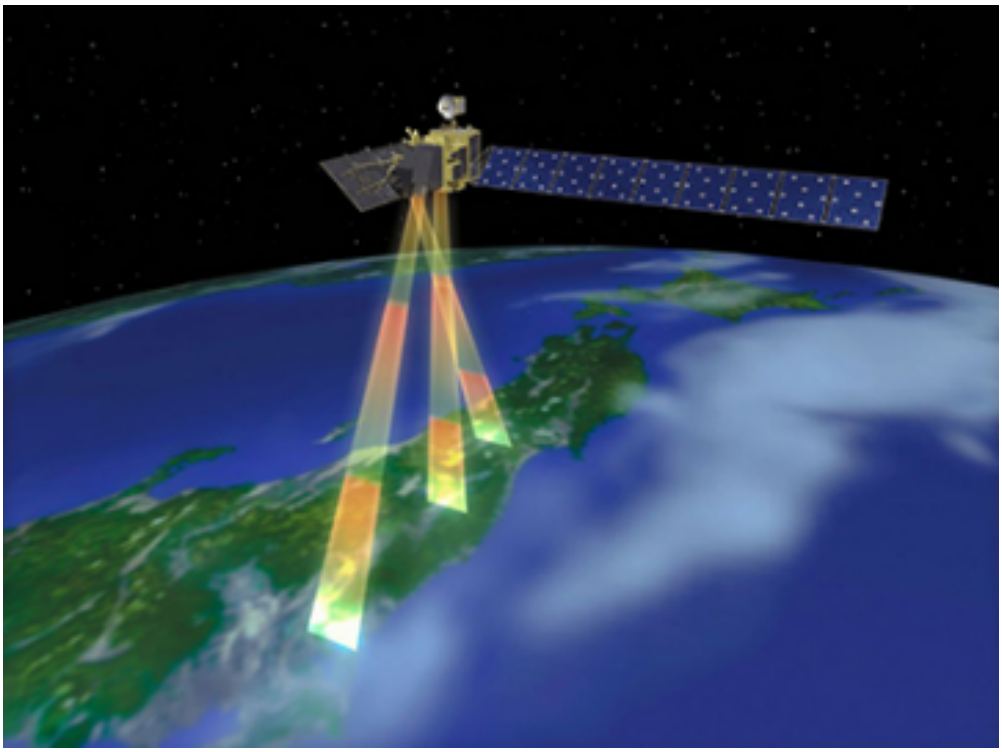

図 1　JAXA 陸域観測技術衛星「だいち」

(画像は JAXA のウェブサイト [1] から引用)

図 2　ユビキタスブイ

(画像は [2] から引用)

我々はリモートセンシングを漁業に利用することで衰退してきている第一次産業に従事している人を増やすことなどを目的としている。この漁業の分野でリモートセンシングを利用する例はいくつか存在するが、これらはいずれも地上のデータなどの不必要な情報も入っているなどの問題点が存在する。よって我々は不必要なデータをなくすことで個人でもリモートセンシングを用いた情報を扱えるアプリを開発することを課題とした。
まず、リモートセンシングとは、主に人工衛星を用いて X 線、紫外線、赤外線などの電磁波を観測し、直接物に触れずにさまざまな情報を知る技術である。リモートセンシングはこのような特性を生かして、ヒートアイランド現象の問題や漁場予測、天気予報などに役立っている。具体例として 2011 年 3 月 11 日に起きた東北地方太平洋沖地震により受けた津波の被害をリ

モートセンシングにより把握することができている。地震調査研究推進本部では「広大な津波浸水域の空間分布を把握するために、JAXA 陸域観測技術衛星「だいち」(図 1) の光学センサ画像（2011 年 3 月 14 日に撮影）を解析しました」[1] とあり津波による被害を確認したり、「国土地理院による直下視・斜め視空中写真を判読したりして、建物被害と建物構造種別の分布を明らかにしました。写真からの判読なので、特に構造種別の判読精度に限界はありますが、発生直後の緊急観測により直下視・斜め視の空中写真を取得できれば、迅速な被害把握に活用できることを実証しました。」[2] といった建物被害の把握をすることができている。これらの成果より、効果的な災害救援活動や復旧活動を行う事ができている。

キーワード　リモートセンシング, 人工衛星, 漁業

（※文責: 飯田珠羅）

Abstract

We aim to increase the number of people engaged in the declining primary industry by using remote sensing in the fishing industry. There are some examples of using remote sensing in this field of fishing, but all of them have problems such as containing unnecessary information such as ground data. Therefore, we set the task of developing an application that allows individuals to handle information using remote sensing by eliminating unnecessary data.
First ,remote sensing is a technology that mainly uses artificial satellites to observe electromagnetic waves such as X-rays, ultraviolet rays, and infrared rays, and to know various information without directly touching objects.
Remote sensing makes use of these characteristics and is useful for problems such as the heat island phenomenon, fishing ground forecasts, and weather forecasts.
As a specific example, the damage caused by the tsunami caused by the 2011 off the Pacific coast of Tohoku Earthquake on March 11, 2011 can be grasped by using remote sensing.
The Earthquake Research Promotion Headquarters said “They analyzed the optical sensor image (taken on March 14, 2011) of the JAXA land observation technology satellite "Daichi" (Fig.1) in order to understand the spatial distribution of the vast tsunami inundation area” [1] and confirmed the damage caused by the tsunami. In addition, the Headquarters for Earthquake Research Promotion "We clarified the distribution of building damage and building structure types by reading aerial photographs of direct and diagonal views by the Geographical Survey Institute. Since it is a reading from photographs, there is a limit to the reading accuracy of each structure type, It has been demonstrated that if aerial photographs of direct and oblique views can be obtained by emergency observation immediately after the outbreak, it can be used for quick damage grasping. ” [2] It is possible to grasp building damage. Based on these results, effective disaster relief activities and recovery activities can be carried out.

Keyword Remote Sensing, Satellite, Fishing

（※文責: 飯田珠羅）

目次

第 1 章　背景

1.1　該当分野の現状と従来例

現在、第一次産業は全体的に衰退してきている。第一次産業就業者数を見ても、1951 年には 1,668 万人いた農林漁業者数は、2020 年には 213 万人にまで減っている [3]。函館市も平成 15 年は漁業就業者数が 4099 人、漁業経営体数が 2159、平成 20 年に 3657 人、漁業経営体数が 1908、平成 25 年には 2959 人、漁業経営体数が 1629、と年々漁業就業者数と漁業経営体数が減っている状況である [4]。

漁業が盛んな地域として北海道の釧路港、青森県の八戸港、宮城県の石巻港、宮城県の気仙沼港、静岡県の焼津港、鳥取県の境港があげられる。函館市も漁業が盛んな地域としてあげられる。函館市は海に面した街ということもあり、漁業が盛んである。具体例として、イカ漁やナマコ漁がある。そして、IT によって漁業を効率的にする取り組みも進んでいる。代表的な例として、”ユビキタスブイ”(図 1.2) という機器や”マリン IT”という研究、”デジタル操業日誌”などがある。ユビキタスブイとは 、水産× IT によると「水温などを計測するセンサ、データ送信部、電力源を海洋ブイに設置したものです。従来から存在する同様の装置は大型、高コスト、専用回線など、研究使用色の強いものでありましたが、ユビキタスブイは小型軽量、低コストで、ランニングコストについても低価格であり、導入障壁を非常に低いものにしました。また、水温センサは 1 つのブイにつき複数の水深の水温が測定できる多層式であるため、多点多層での水温データの取得が可能となっています。」[2] とある。

この機器を用いて水温などの海の情報を蓄積し、海の状態を把握することで、ホタテ養殖の生産量向上に繋げることができる。

マリン IT とは漁業者の協力を得て漁船でデータを収集してそのデータを用いた研究成果を漁業者にフィードバックすることで、研究開発と水産業の進展を循環させる [5] という研究である。これを研究することが出来れば、漁に出かける前に、どんな魚がどれだけ獲れるかのデータが共有できるようになれば、漁業者はもちろんのこと、物流や加工、販売に至るまで、ビジネスのあらゆる面で効率化が図ることができる。

デジタル操業日誌とは、漁師さんが船上で操業時間や漁獲量を iPad に入力することで、リアルタイムにそれぞれの漁船の情報を共有することができ、これを用いることで計画的な量を行うことができるというアプリであり、結果としてナマコの取りすぎなどを防ぐことと漁獲量を安定させることが出来た [6]。

他にも、marine PLOTTER という船舶の位置情報を共有する iPas アプリが存在し、これを利用することで、一日の漁獲量と併せて振り返ったり、昔から存在する漁師の勘をより分かりやすくデータとして残い、後代に継承することが可能になった [6]。

そこで、本グループは地上のデータなどの不必要なデータの表示をなくし、衛星の専門家や漁業関係者以外の人でも、個人でも衛星データを扱うことが出来るように海面温度や魚のデータをまとめてみることのできるようなアプリを開発することと、釣り初心者でも扱えるように釣具屋に関わる情報や経路などを表示できるようにすること、そしてそのようなアプリを開発した際に魚の漁獲量の増加に関わる社会的責任について考察することを課題とした。

（※文責: 山本陽平）

1.2　目的

我々は上記の通りの衰退してきている第一次産業に従事している人を増やす、または漁業の効率化によって漁業に携わっている人の負担を減らすことが良いと考えた。一度に広い範囲を計測できるリモートセンシングを用いて海を観測することで、函館で盛んな漁業に貢献できるのではないかと予想した。リモートセンシングを用いて、時期ごとの気温、海面温度などの情報を取得し、そこから目的の魚が多く生息する場所を予測した結果を表示する iOS アプリを制作することで、釣りや漁の効率を良くすることを目的とし、活動を行った。

（※文責: 飯田珠羅）

1.3　従来例

漁業分野でのリモートセンシングの利用例がいくつか存在する。リモートセンシングを用いて海面水温分布を算出し、データを解析し、漁業の効率化と生産性向上に役立っている。さらに自動的に毎日同じ地域の観測を行うことができ、海洋環境の季節的、経年的変動を認知するのにも役立っている。しかしこれらの例に関して課題がある。経済産業省によると「政府開発衛星に関して、技術開発面を重視してきたため、ビジネスへの活用、利用側への考慮という視点が衛星を開発する側に欠けている。衛星のスペックに関して、データの継続性を含め、利用側のニーズを汲み取れていない。観測頻度が不十分であるため、利用者のニーズに応えられていない。」とある [7]。これらより現時点では、ほとんどの企業がリモートセンシングを活用することが出来ていない、つまりは衛星システムの企画から運用、データ利用を考えている人が少ないということが予想される。

（※文責: 山本陽平）

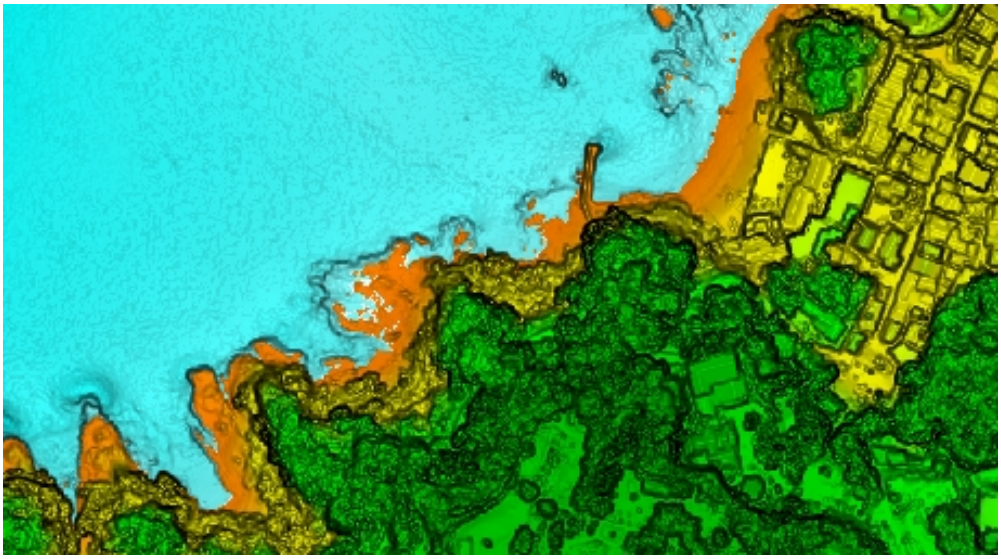

図 1.1　地上のデータが含まれている衛星データ

（画像は [8] より引用）

1.4　従来例の問題点

上記の従来例では東京都や大阪府などの一部の都道府県では海面温度や魚の生息情報などのデータが描かれていたが、その他の都道府県では一切の情報がないという問題点がある。また地上のデータなど、不必要なデータなども乗っているという問題点もあり、そして衛星の専門家や漁業関係者がデータを活用する方法としては良いのかもしれないが、個人で衛星データを活用することは非常に困難であるという問題もある。

（※文責: 飯田珠羅）

第 2 章　到達目標

2.1　問題の設定

本グループの目標はリモートセンシングを用いて取得したデータから魚が多く生息する場所を推定するアプリケーションを作成することである。この目標を達成するために、本グループでは次の問題を設定し、その解決を目指した。まず、リモートセンシングを用いてどのように魚の居場所を推定するかという問題である。また、前節で述べたように都道府県によって海水面温度の情報や魚の生息情報の提供に偏りがあるうえに、提供されているデータには魚の生息域の探知に不必要なデータが混じっているために活用するのが非常に難しいという問題である。

（※文責: 菅原慎哉）

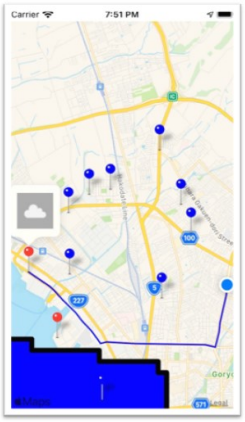

図 2.1　アプリの画面

2.2　課題の設定

前節で述べた問題を解決するために、以下の条件を設定し、それを元にシステムを構築することを課題とした。

・誰でも簡単に衛星データを利用して生産性の向上などに役立てることができる

・衛星データをわかりやすく利用者に伝えることができる

・利用者のニーズに合ったデータを提供できる

以上を踏まえた結果、本プロジェクトでは漁師、釣り人、観光客を対象とした iOS アプリの開発を目指すこととした。アプリ開発するにあたって、グループでその手順を考察し、課題と課題解決のための必要技術、知識を以下のように設定した。

・QGIS による衛星データの解析（データ解析の知識、Python）

・クラウドサーバー（AWS での EC2 サービス）、UNIX コマンドや UNIX でのファイル操作の知識、およびにデータベースの構築（サーバーおよびデータベースの知識、SQL）・アプリの機能、レイアウト設計（Swift）

（※文責: 外川雄也）

2.3　課題の割り当て

本グループではフロントエンドチームとバックエンドチームの大きく 2 つのグループに分かれて、それぞれ別々の作業を行い、目標の達成を目指した。Swift でのプログラムの作成や UI の作成はフロントエンドチームで行った。また、解析元となるデータの入手から QGIS での画像解析や、サーバーの構築はバックエンドチームで行った。

（※文責: 飯田珠羅）

2.4　到達レベル

バックエンドチームで作成したプログラムによって人工衛星を用いたリモートセンシングから函館付近の海面温度のデータを入手し、そのデータは Python を利用し、QGIS を通して解析を行う。解析したデータはサーバを通してフロントエンドチームアプリ側に送信する。データより旬の魚や地域の魚の位置情報を魚の特性や生息地域をもとに予想する。予想して得た情報は Swift を使い iOS アプリにして地図表示 (図 2.1) をする。

（※文責: 山本陽平）

2.5　解析手法

前節でも述べたとおり、インターネットで提供されている衛生の地表面データを元に最終的には魚のいる場所を推定する方法についてチームで検討した。最終的な方法としては次のようになる。まず、日本周辺の海において冷たい海流と温かい海流がぶつかるところがある。ここは潮目と呼ばれていて、植物プランクトンが集まりやすく、それに伴って魚も集まりやすいことが知られている。本グループでは海水面の温度データを解析して温度変化の激しい場所を特定し、それを元に魚のいる場所が推定できる事がわかった。

（※文責: 外川雄也）

2.6　活動の進め方

2.6.1　システムの構成

システムのおおよその構成は次のようになる。

1. G-Portal から、解析する元となる函館周辺の海水面温度のデータを入手する。
2. 入手したデータから海水面温度変化の激しい場所を特定し、その激しさの度合によって色分けする。
3. 色分けしたデータの不必要なデータを廃棄し、データを軽量化する。
4. データを iOS アプリで表示できる形式に変更する。

5. データを AWS の S3 ストレージに保存し、Swift 側でそのデータにアクセスして保存する.
6. Swift 上で実際に画面に表示される UI やアイコン、その他必要な機能を設計する。

（※文責: 外川雄也）

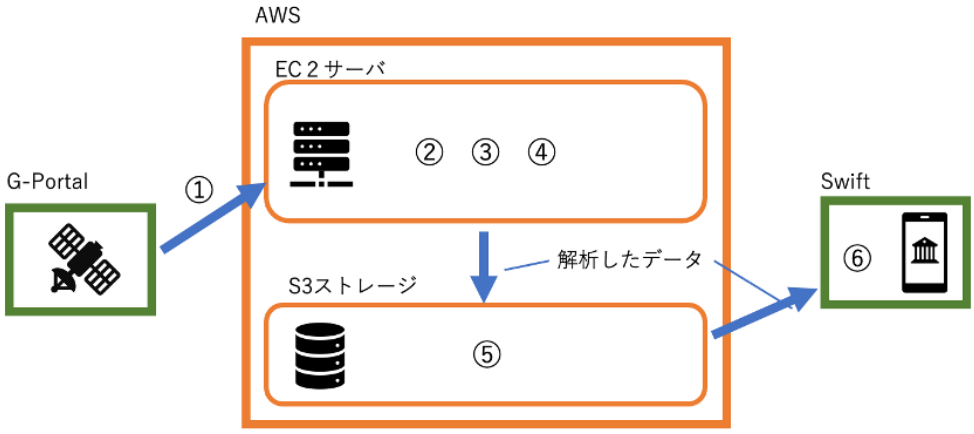

図 2.2　システムの構成

2.6.2　システムの構成の詳細

データ取得には G-Portal を利用し、解析には Python と QGIS を使用する。G-portal とは、JAXA が運営している地球観測衛星提供システムである。また、QGIS とは、オープンソースで使うことが可能なソフトウェアで、地理空間情報の作成や編集、分析ができる [9]。まず、AWS を用いて構築したレンタルサーバ内で Python によって G-portal からデータを取得する。次に、サーバ内に取得したデータを Python と QGIS を用いて日本全土及び近海を海水温度毎に色を分ける（図 2.3）。そして、色が分けられたデータを XYZtiles というファイル形式にし、サーバ内に保存する。XYZtiles は国土地理院が配信するタイル状の地図データのことを指す [10]。アプリの制作にはレンタルサーバに入れた Python によって解析されたデータとインターネットから取得した函館の魚の情報をまとめたものを利用する。また、函館の旬の魚の情報を地図表示したものを Swift を用いて実装し、iOS アプリで表示できるようにする。また、アプリで表示する際に利用者が使いやすいような UI にするための学習を行い、そのためのデザインの学習を行う。学習にはインターネットに存在する記事を読んだり、出版されている書籍を読んだりして学習する。

（※文責: 外川雄也）

2.7　技術の習得

開発をするにあたって、必要な技術が多く、夏季休暇中にこれらを習得することを目指した。必要な技術は下記の４つである。

・QGIS
・Swift/Xcode
・AWS
・Python

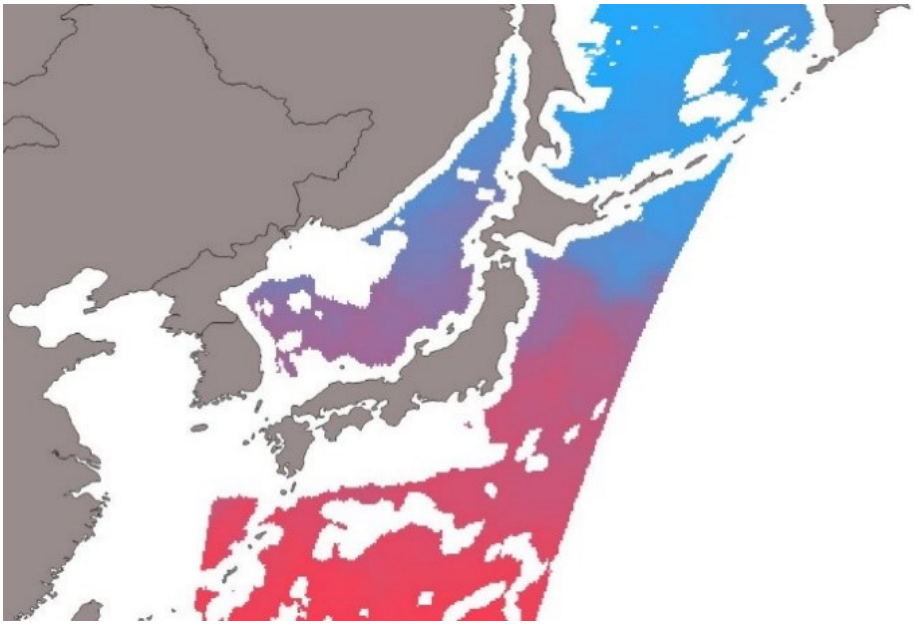

図 2.3　システムの構成の詳細

2.7.1　QGIS

QGIS はオープンソースソフトウェアの GIS ソフトであり、地理情報データを閲覧、編集、分析することができる。また、プラグインと呼ばれる拡張機能も無料で使うことができる。ほかにも GIS ソフトは ArcGIS などがあるが、無料で使えて世界中で広く使われているため、文献も多いことから、本プロジェクトでは GIS ソフトとして QGIS を採用した。

2.7.2　Swift/Xcode

2014 年に Apple 社が発表した iOS や Mac 向けのネイティブアプリケーションを開発するためのオープンソースのプログラミング言語である。Apple の iOS 7 以降、OS X version 10.9 以降の OS を搭載しているデバイスのすべてのアプリケーション （Apple TV、Apple Watch など）を Swift で開発することができる。

2.7.3　AWS

AWS とはアマゾン・ウェブ・サービスの略で、アマゾンが提供している様々なクラウドコンピューティングサービスの総称である。AWS では、様々なサービスが提供されているが、本プロジェクトでは、サーバーを提供している Amazon EC2 とストレージサービスを提供している Amazon S3 を使用する。

2.7.4　Python

バックエンドチームでの解析の処理や、サーバーでの処理を実行する際のプログラミング言語として、Python を採用した。理由としては、QGIS では Python 用のコンソールが標準で用意されていて、そのコンソール内で、QGIS の Python 用の API である PyQGIS を利用するためである。

（※文責: 外川雄也）

2.8　開発手法

本プロジェクトでは前節で述べた通り、フロントエンドチーム 3 人とバックエンドチーム 3 人の 2 つのグループに別れて、分業体制で開発を進めた。フロントエンドチームでは Swift を用いて iOS アプリの開発をした。1 人のパソコン（メインパソコン）ですべてのコードを書いていき、他の人が書いた Swift のプログラミングコードをチャット等で共有してもらい、メインパソコンにコピペした後に、必要に応じて修正する方法を用いて開発を進めた。一方、バックエンドチームでは更に、QGIS チームとサーバーチームの 2 チームに別れて作業した。QGIS チームは 2 人で構成されていれ、画像解析ソフトである QGIS を用いて、海水面温度から魚の位置を推定した画像を表示する方法の検討を行った。サーバーチームは 1 人で構成されていて、主に AWS を用いた、サーバーの作成や、サーバーでの環境構築を行った。

（※文責: 外川雄也）

第 3 章　前期活動内容

3.1　前期の目標

前期の目標として大きく 5 つを挙げた。

1 つ目は言語についてである。まず初めに使用する言語をグループで話し合った。フロントエンド側では iOS アプリ制作を作りたいという声があったため言語は Swift、バックエンド側ではサーバの管理に使いやすくかつインターネット上に多く情報が載っている Python を採用した。よって Swift の習得、Python でのサーバ管理方法の習得を目標にした。またプロジェクトの予算より、Swift についての解説本や iOS アプリ設計パターン入門を購入し、プログラムを作成する際に問題が発生した際にスムーズに解決するために使用した。そのほかに YouTube のプログラミング講座を利用した。

2 つ目はバックエンドについてである。バックエンドで解析した画像をフロントエンドで表示する際、容量が大きすぎるため表示するまでにかかる時間が長いという問題に直面した。そこでバックエンド側からアプリにデータを提供する際の負荷を軽減する方法の勉強をし、上記の問題を解決することを目標にした。

3 つ目はアプリの内容構成についてである。フロントエンドの作業を円滑に進めるために釣り人や漁師などの利用者の目線に立ち、ほかの海アプリも参考し、必要である機能を Google Jamboard にフロントエンドのメンバーがアイディアを書き出す。これを数回行い改善することを目標とした。

4 つ目はプログラミングの進行方法についてである。それぞれのプログラミングをするうえでどのように進めていくのかを Google Jamboard や Teams で書き残し、今後の活動をスムーズに進めることを目標にした。フロントエンドでは上記の 3 つ目の目標からアイディアを出し、アイディアごとに役割を決め、実際にプログラムを制作していった。バックエンドではサーバを仮設立するためにインターネットを活用し技術を習得する役割と G-portal からダウンロードしたデータを解析するための周辺知識を取り入れる役割や、解析するための計算方法を考える役割に分かれて活動した。

（※文責: 山本陽平）

3.2　活動過程

Group A では前期では 5 月から 6 月上旬までにかけてはフロントエンドとバックエンドが共通の活動を行い、6 月から 7 月までにかけてはフロントエンドとバックエンドで別々の活動を行い、また 7 月から前期終了までにかけてはフロントエンドとバックエンドで共通の活動を行った。

共通して行った制作活動過程は以下のとおりである。

5 月上旬　リモートセンシングについての学習。
5 月下旬　制作物についての企画。
7 月上旬　ポスターや発表用スライドなど、提出物作成。

7 月下旬　前期報告書作成。

フロントエンドの制作活動過程は以下のとおりである。

6 月上旬　Swift 環境構築。
6 月下旬　Swift の学習, UI デザインの学習。

バックエンドの制作活動過程は以下のとおりである。

6 月上旬　Python 環境構築。
6 月下旬　Python の学習, サーバ構築の学習。

（※文責: 飯田珠羅）

3.2.1　フロントエンドの活動詳細

6 月上旬　Swift の環境構築を行いつつ、ユーザーが使いやすいデザインがどのようなものであるかどうかを、既存例を参考にインターネットで検索し学習した。 また利用者は解析結果とともにどのような情報を求めるかをみんなで話し合い、アプリの大まかな要素を決定した。
6 月下旬　参考資料を中心に Swift で変数の取得方法やハンバーガーメニューなどを学習し、アプリに実装するなど、実際に個人で本やサイトなどを参考にしながら一人一つアプリを作成することでアプリのデザインの決定及び swift の勉強を行い、本などの情報を見ることでコードがどのように動いているかをだいたい理解できるような状態にした。

（※文責: 飯田珠羅）

3.2.2　バックエンドの活動詳細

6 月上旬　QGIS で行う画像解析を Python で行うために、インターネットにある様々なサイトを参考にし、学習した。またどのサーバを用いるかを、いろいろなサーバから候補を選び出しその中から実際に用いるサーバを決定した。
6 月下旬　h5py や OpenCV などの様々な画像解析を行う方法について学習し、サーバからアプリにデータを提供する際に負荷が少ない XYZtiles という出力方法を学習することができた。また、練習用のサーバを構築し、サーバを動かすテストを行った。

（※文責: 菅原慎哉）

3.2.3　担当課題と他の課題の連携内容

アプリがサーバから解析された画像を表示するときに、サーバからデータを提供してもらうまでの間は利用者にしばらく待機してもらう必要がある。また、アプリに表示するまでにかかる負荷と利用者が待機しなくてはいけない時間はファイルの大きさによって変化すると考える。そのため、フロントエンド側とバックエンド側でアプリ側と利用者にあまり負荷・負担をかけないようにするための作業や話し合いを連携して行った。

（※文責: 山本陽平）

3.3 前期の成果

新型コロナウイルスの影響により、授業時間の 3 分の 1 しか対面で活動することが出来ず、プロジェクトでのコミュニケーションをとることが難しかった。そのため先生方を含め、メンバー全員でコミュニケーションをとるために Zoom や Teams を利用した。前期でできた成果としてバックエンド側ではプロジェクトの開始後から始めていた Python で G-portal から海面温度などのデータを得ることが出来たことや、手動でデータを得られることが出来たこと。フロントエンド側では swift を用いて画面遷移や地図の表示などの iOS アプリの最低限の機能や追加機能を決定することができたこと。旬のお魚の検索機能をマップ上で使用可能にする、タグ検索を可能にすることが追加機能の案として決定したこと。　両方で行ったこととして得られたデータを XYZtiles のファイル形式 にすることによって、サーバ側からアプリにデータを提供する際の負荷を軽減することができたことがそれぞれ挙げられる。

（※文責: 菅原慎哉）

3.4 中間発表会まとめ

中間発表会は 7 月 9 日に行われた。発表形式はオンラインで Zoom を使用した。当日 15 時より各プロジェクトの報告用動画、報告用ウェブサイトの公開が開始された。16 時より前半質疑応答を１５分間し、５分休憩のち 2 回目の質疑応答を開始した。この質疑応答を前半 3 回、後半３回ずつ行った。質疑応答する際、グループを２つに分け、一つのグループが質疑応答にあたっている間はもう一つのグループが事前にプロジェクトメンバー内で決めた他のプロジェクトの評価に回った。質疑応答中は Zoom の機能である画面共有を行い、過去に質問された内容とその回答などを表示しながら質疑応答を行った。

中間発表から以下の質問をいただいた。「⇒」は質問に対する返答である。

・魚を特定することができるのか。

⇒特定まではできない。魚の生息を解析して多くいそうなところを地図に示す。

・魚の位置や動きなどの規則性の特定はどうやる?

⇒調べた生態と海面温度の関連付けによって特定。

・船に搭載されているレーダーと比べて優位性はあるのか?

⇒船釣りよりも一般の釣り人向け。

・ G-Portal からのデータの取得方法。

⇒ WinSCP で G-Portal に接続してデータを取得する方法と G-Portal から手動でデータを取得する方法の二つがある。

・釣りや漁を効率化すると、魚の取りすぎになって、持続可能な社会と矛盾する気がするのですが、どう考えますか。

⇒集団に対してではなく個人に対してのアプリなので影響はすくない少ないと考えられる。ただもしものために何か対策を考えておこうと思う。

・アプリリリースの社会的責任は？

⇒使用用途は基本的に個人に任せ、想定されるラインの設定などはこれから検討。

・過去のデータを確認して魚の有無を確認するのか、センサーを用いて確認するのか？

⇒リモートセンシング（衛星データ）を用いて快音などを用いて調べます。センサーではないです。100 %ではなくあくまで予測です。

・海の上で使うことも想定していますか。

⇒近海も想定しています。通信衛星を用いることで海の上でも利用できるようにする予定です。

（※文責: 菅原慎哉）

3.5 新たなる課題

前期の成果や中間発表からいただいた質問より後期の課題として 4 つの課題が生まれた。

1. 中間発表の質疑応答の時間に参加者からアプリリリースの社会的責任について予想されるラインの設定について質問され、想定してなかったため、アプリをリリースする際、社会的責任について農林水産省やアプリによる漁獲量を予想し、基準を設け、その内容を利用規約として掲載することを後期の課題とした。
2. 既存のアプリや船に備わっている機械とのすみわけとして、追加機能として天気の表示や釣具屋までの経路表示などを考え、ほかのアプリにはない機能を入れる事を課題とした。
3. 前期の成果でも書いてあるように、旬のお魚の検索機能をマップ上で使用可能にすることやタグ検索などの新規機能の追加を行うことを後期の課題にした。
4. 練習用の仮サーバを構築し、起動することには成功したが、本サーバを構築することはまだ行っていないため、本サーバを構築し、データを本サーバで活用したりすることが出来るかを確かめたりフロントエンド側のアプリと接続してデータを送ることが出来るかを確かめることを後期の課題にした。

（※文責: 飯田珠羅）

第 4 章　後期活動内容

4.1　後期の目標

後期の目標として大きく 6 つを挙げた。

1 つ目はフロントエンドとバックエンドの連携についてである。前期ではフロントエンドとバックエンドの 2 つのグループに分かれて活動を行っていたが、後期からは 2 つのグループで連携を行い、バックエンド側で得られたデータをフロントエンド側で表示できるようにしなくてはいけないためこれを目標とした。

2 つ目はアプリをストアで配信することである。プロジェクトの Group A内でアプリ完成の目安としてふさわしいものは何かと考えた時に、アプリを Apple store で配信することだという結論が得られた。よってアプリをストアで配信することを後期の目標とした。

3 つ目は利用規約を書くことである。参加者からアプリのリリースに伴う社会的責任について予想されるラインの設定について質問されたが、前期終了時点では想定してなかったためアプリをリリースする際、社会的責任について農林水産省やアプリによる漁獲量を予想し、基準を設け、その内容を利用規約として掲載することを後期の課題とした。

4 つ目は既存のアプリとの差別化のための新規機能の追加についてである。これは、前期終了時点でのアプリの機能が画面遷移や地図の表示などの最低限の機能しか実装することが出来ていなかったため、旬のお魚の検索機能をマップ上で使用可能にすることやタグ検索などの新規機能、及び既存のアプリや船に備わっている機械との棲み分けとして天気の表示や釣具屋までの経路表示などの追加機能を実装することを後期の目標とした。

5 つ目は本サーバの設立についてである。前期終了時点では練習用の仮サーバを設立することが出来ていたが、本サーバを設立することをまだ行っていなかったため、これを後期の目標とした。

6 つ目はバックエンド側のデータ解析の自動化である。前期では G-Portal からデータを受け取り解析するまでの一連の流れを手動で行っていたため、このままアプリとしてフロントエンドとバックエンドで連携する際には 1 日 1 回早朝に起きてデータの取得を行わなくてはいかなかった。そのため、データの取得及び解析を自動化することを後期の目標とした。

（※文責: 飯田珠羅）

4.2　活動過程

GroupA では 8 月からは、海グループは再度開発・解析を行い、都市グループは言語学習やデータ収集、開発を行った。11 月後半には、成果発表に向けた準備を行った。

共通して行った制作活動過程は以下のとおりである。

9 月下旬　　今後の予定の確認
10 月　　特になし
11 月　　ポスターやスライドなどの提出物作成
12 月　　最終発表、グループ報告書の作成

フロントエンドの制作活動過程は以下のとおりである。

9 月下旬　地理情報付きの画像をマップに表示させる方法の模索、QGIS で解析した情報をアプリに反映

10 月上旬　指定した位置の天気情報の割り出し、UI の見直し、現在地から目的地の経路を求める方法の模索

10 月下旬　位置情報の取得、釣り具屋の経路を求める機能の ON・OFF の実装

11 月上旬・下旬　プログラムの統合

バックエンドの制作活動過程は以下のとおりである。

9 月下旬　サーバの構築、データ変換や解析

10 月上旬　サーバへのデータ保存、データ変換、データの色付け

10 月下旬　ラスタのベクタ化を行う Python プログラムの作成

11 月上旬　サーバにおける環境構築、QGIS の画像解析に関して、画像のサイズを小さくする、ファイルの自動ダウンロード、QGIS 上でデータに色付けするための Python プログラムの作成、ベクタレイヤのマスク処理をする Python プログラムの作成

11 月下旬　サーバ内での Python の自動化、最新の h5 ファイルを取得する手順の自動化

4.2.1　フロントエンドの活動詳細

9 月下旬　Swift を用いて地理情報付きの画像をマップに表示させる方法を模索した。また、QGIS で解析した海面温度の情報をアプリに反映させた。

10 月上旬　指定した位置の天気情報の割り出し・表示をする方法や、現在地から釣り具屋などの目的地の経路を求める方法を模索した。また、UI の見直しを行った。

10 月下旬　位置情報を取得し、天気情報や経路を実際の位置情報を用いて表示した。また、釣り具屋の経路を求める機能を ON・OFF で切り替えることができるように実装した。

11 月上旬　サーバにおける環境構築、QGIS の画像解析に関して、画像のサイズを小さくする、ファイルの自動ダウンロード、QGIS 上でデータに色付けするための Python プログラムの作成、ベクタレイヤのマスク処理をする Python プログラムの作成

11 月上旬下旬　これまでに作成したプログラムを統合し、アプリが正常に動作するようにした

4.2.2　バックエンドの活動詳細

9 月下旬　バックエンド側は前期に引き続き、サーバの構築などを行うチームと QGIS での解析をするプログラムを書くチームの 2 つに分かれて作業した。 サーバの構築などを行うチームでは、S3 サーバに接続することを目標とし、活動を行った。 QGIS チームは、G-Portal を用いてダウンロードした h5 ファイルを GeoTIFF ファイルに変換することを目標にして活動を行った。

10 月上旬　サーバでのデータの保存を行った。また、h5 ファイルから GeoTIFF ファイルへの変換、GeoTIFF ファイルから GeoJSON ファイルへの変換を行う方法を模索した。また、海面温度の温度差による傾斜を表示し、傾斜の大きいものに色付けをする Python プログラムを作成した。

10 月下旬　10 月上旬に行った GeoTIFF ファイルから GeoJSON ファイルへの変換などを自動的に行うために、Python プログラムによってこれらの動作を行えるようにした。

11 月上旬　サーバにおける環境構築を行った。また、G-Portal のファイルを自動ダウンロードする Python プログラムの作成、QGIS 上でデータに色付けするための Python プログラムの作成、ベクタレイヤの不要な部分を取り除く処理をする Python プログラムの作成を行った。

11 月下旬　これまでに制作した Python プログラムをサーバ上で自動的に実行するようにした。また、G-Portal にある最新の h5 ファイルを自動でダウンロードする Python プログラムの作成を行った。

（※文責: 菅原慎哉）

4.3　成果発表まとめ

成果発表会は 12 月 10 日に行われた。発表形式はオンラインで Zoom を使用した。当日 15 時より各プロジェクトの報告用動画、報告用ウェブサイトの公開が開始された。16 時より前半質疑応答を１５分間し、５分休憩のち２回目の質疑応答を開始した。この質疑応答を前半３回、後半３回ずつ行った。質疑応答する際、グループを２つに分け、一つのグループが質疑応答にあたっている間はもう一つのグループが事前にプロジェクトメンバー内で決めた他のプロジェクトの評価に回った。質疑応答中は Zoom の機能である画面共有を行い、過去に質問された内容とその回答などを表示しながら質疑応答を行った。しかし、前半質疑応答中、ブレイクアウトルームに入れなくなる Zoom のトラブルが発生し、前半の質疑応答の時間が 5 分ずれた。前半から後半に移るまでの間の 10 分休憩が 5 分休憩に変更され、後半質疑応答はスケジュール通りに進行された。

中間発表から以下の質問をいただいた。抜粋した質問を掲載する)「⇒」は質問に対する返答である。

・それぞれの班で努力を考えないとして、解決したかった問題は？

⇒フロント：マップを表示する機能を作成するのに時間がかかった。もう少し短い時間で解決したかった。

バック：サーバで QGIS は動き、処理は可能だった。だが、自動化するときに Ubuntu でのリポジトリの取得が難しかった。もう少し簡単に取得したかった。

・釣り場のデータは川釣りでも応用できますか？

⇒川は表示されない。大きな湖ならいけるが、小さい川では難しい。

・川の温度をデータとして使えば、いけるのか？

⇒衛星からのデータがリアルタイムで取得できれば可能である。

・海のデータを取るのは、1 ピクセルあたりどのくらいの情報（精度）なのか？？

⇒だいたい 200 mである。

・作業中楽しかったことは？

⇒フロント: プログラムを書いている時。

バック:QGIS で解析するコードを書いていて、エラーが治ったときやエラーの無いコードが書けたとき。

・想定しているユーザーは誰か、ユーザーは何を得られるのか。

⇒釣り人を想定している、魚の釣りやすい場所が分かる。

（※文責: 山本陽平）

4.4 新たなる課題

後期の生かや成果発表からいただいた質問より今後の課題として 5 つの課題が生まれた。

1. アプリを作る際に立てた最初の目標である「アプリをストアで配信すること（または審査を申請すること)」を達成することが出来なかったため、これを今後の課題にした。
2. 解析を行った画像をアプリ上に自動で更新するためのプログラムを作成することが出来ず、手動で行う必要があるため、この自動化を行うためのプログラムを作成することを今後の課題とした。
3. 今回のプロダクトは海に焦点を当てて作業を行ったため、海だけの情報のみを扱うことになり、川の情報がない。そのため川の画像をバックエンドにより解析、川の情報を追加することを課題とした。
4. 今回の海グループのプロジェクトは海面水温を解析し、魚の釣り場所をアプリ上に表示した。しかし、単一の情報だけでは解析精度に欠けると考えられる。よってより解析精度を上げるため、海面水温だけでなく、クロロフィル濃度を解析することで、より正確に魚の釣り場所をアプリ上に表示することを課題とした。
5. アプリを使う際に検索機能や旬の魚の釣り床の表示ができていないので UX の改善、利用者が使いやすいように UI の改善を課題とした。

（※文責: 山本陽平）

第 5 章　結果

5.1　プロジェクトの成果

プロジェクトの成果は、フロントエンドとバックエンドそれぞれ以下のとおりである。

5.1.1　全体

・アプリ作成

5.1.2　フロントエンド

・解析した画像をアプリ画面の表示
・機能の追加（天気、目的地の経路、釣り具屋の表示、位置情報）

5.1.3　バックエンド

・サーバの作成
・画像の取得、解析の一連の流れの自動化

（※文責: 飯田珠羅）

5.2　成果物

本プロジェクトの GroupA は漁師や釣り人が使いやすいアプリを制作した。下の図ｎが実際のアプリの画面である。本アプリには多くの釣りのアプリにある必要な機能である魚の釣りやすい位置を表示のほかに、4 つの機能を追加した。1 つ目は現在位置を表示する機能だ。図 5.1 の水色の丸がそれに該当する。2 つ目は函館の釣具屋を表示し、名前を見ることができる機能だ。これは図 5.1 の青いピンが該当する。3 つ目は目的地を選択し、経路表示をする機能だ。これは図の水色の丸と赤いピンをつなげている青い線が該当する。4 つ目は指定した地点の現在の天気を表示する機能である。これは図 5.1 の赤いピンの上にある雲の画像が該当する。

（※文責: 飯田珠羅）

5.3　解決手順

フロントエンドとバックエンドの解決手順はそれぞれ以下のとおりである。

5.3.1　フロントエンド

機能アイディア　Google Jamboard にてアイディアを書いた。アイディアを書く際、注意しなければならないのはアプリのコンセプトに関係ない機能を追加してはいけないことである。そ

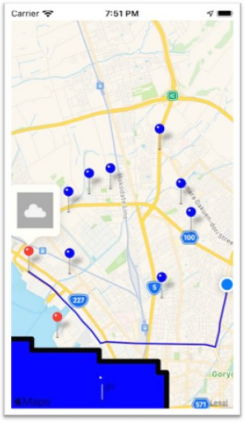

図 5.1　アプリの画面

のために釣りのアプリや釣りに関するブラウザを調査し、共通している必要最低限の機能を書き出していった。そして次に、ほかの釣りのアプリや釣りに関するブラウザとのすみわけを考えた。今回は漁師や主に釣り人を対象としたアプリを制作するので、他のアプリやブラウザの棲み分けとして釣具屋の情報や経路表示などの様々なアプリの機能をアイディアとして出すことにした。

（※文責: 飯田珠羅）

機能の追加　まず初めに、フロントエンド側のメンバーで集まってどのような機能を搭載するかを、実現可能かは置いておいていくつか案を出した。その後ユーザーの立場に立ち、必要性が高い順に優先度を設定し、さらにその中でも実現のしやすさで順序を立ててリスト化を行った。そこからはメインとなる機能を追加していきつつ、実現性の高い機能である天気の表示や目的地までの経路といった機能を追加していった。この機能のうち、天気の情報の表示や経路表示については山田純平が担当を行い、函館市にある釣具屋の表示についての機能は山本陽平が担当を行い、位置情報の表示についての機能は飯田珠羅が担当を行った。

（※文責: 飯田珠羅）

5.3.2　バックエンド

サーバ　バックエンドチームではサーバチームと QGIS を用いた画像解析チームに分かれて作業を行った。サーバは外川雄也が担当し、画像解析は蓬畑尚輝と菅原慎哉が担当した。サーバ側では前期の終わりに、フロントエンドチームと相談しながら、バックエンドチームで G-Portal からのデータの取得からアプリでのデータの出力までのフローを作成し、サーバで実装する機能などを明確にしてそれらの共有をプロジェクト時間に Zoom やホワイトボードを通して行った。後期から本格的に開発を始めた。AWS についての学習には市販の教材とインターネット上の資料を参考にしながら学習を行った。

（※文責: 外川雄也）

画像解析　画像解析チームでは Zoom を通して情報や進捗の共有を行いながら、開発に取り組んだ。主に QGIS で行う海面温度の傾斜の計算やデータの形式変換などの画像解析の方法を

調べ、それらの工程を Python プログラムによって自動的に実行する方法を考えた。蓬畑尚輝は海面温度の傾斜の計算や、計算した傾斜を基準にした画像の色付けの工程に取り組んだ。また、菅原慎哉はデータの解析やフロントエンドとのデータのやり取りを行ううえで必要な形式変換の工程に取り組んだ。これらの取り組みで作成した Python プログラムを蓬畑尚輝が管理し、サーバチームの外川雄也と協力し、プログラムの自動化を実現させるために取り組んだ.。

（※文責: 外川雄也）

図 5.2　解析前の画像

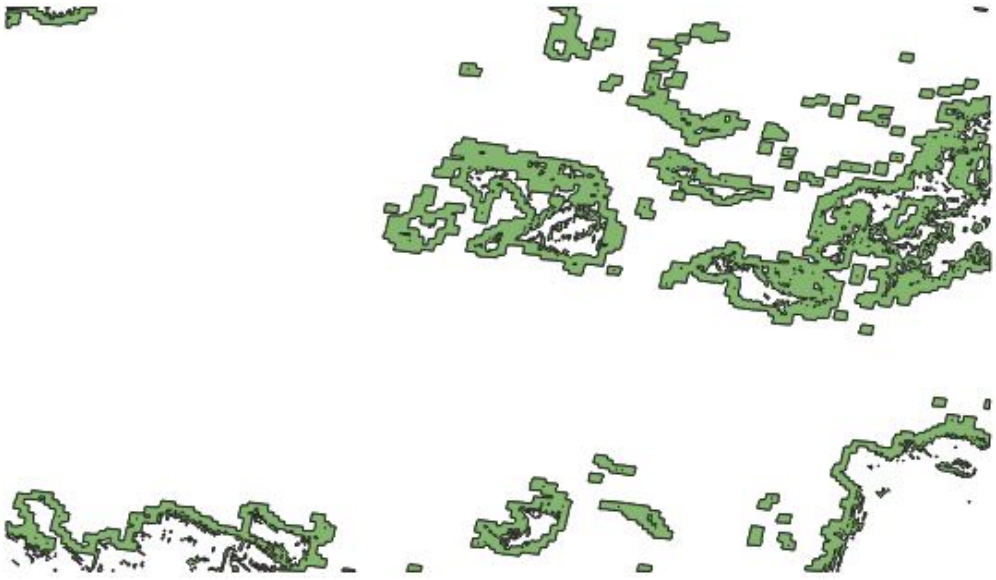

図 5.3　解析後の画像

5.4　未解決課題とアプローチ

フロントエンドとバックエンドの未解決の課題はそれぞれ以下のとおりである。

5.4.1 フロントエンド

利用規約　未解決の課題の一つとしてアプリの利用規約の執筆がある。今回利用規約を書くことが出来なかった理由としてプロジェクトのメンバーが利用規約についての知識がなく、また優先順位も低かったことが挙げられる。なので、この課題を解決するための時間も確保することができなかった。

UI と UX の改善　フロントエンド側は前期に UI・UX について本を読むことによって勉強を行った。だが、それが足りなかったためアプリとして見栄えが良くないものになってしまっていた。よってこれからはより UI・UX について勉強を行うことでこれらを改善していくことが課題である

（※文責: 菅原慎哉）

5.4.2 バックエンド

自動化　開発を始めた当初でのバックエンドチームの目標は G-Portal から取得したデータを解析してアプリがアクセスするサーバのストレージに保存するという機能を定期的に自動で行う予定だった。そのために、手動でデータの取得から画像解析、データの出力までを行い、それを順次自動化していくことを目指した。しかし、「G-Portal のサーバから目標となるデータのみを取得する」という作業と、「サーバで QGIS を用いて画像を解析する」、「サーバで解析したデータをフロントエンド側で受信できるようにする」という作業の自動化ができなかった。まず、「G-Portal のサーバから目標となるデータのみを取得する」という問題について解説する。G-portal から自動で必要なデータを取得するために、Python を使って G-Portal のサーバに SFTP 接続をしてコードで指定したデータ名と同じデータ（最新の函館付近の海水面温度のデータの名前を指定する）を取得するというコードを書くことを目指した。しかしながら、最新の函館付近のデータ名が不規則であり、Python のコードで指定できないという問題が生じた。結局その問題は解決できず、手動にして他の機能の実装に着手することとなった。つぎに、「サーバで QGIS を用いて自動で画像を解析する」という問題について解説する。サーバで QGIS の Python 用の API である「PyQGIS」を用いて Python の画像解析のためのスタンドアロンスクリプトの作成を目指した。しかし、必要な Python 用ライブラリのインポートの部分でエラーが起きてしまう問題が生じた。インターネットの文献を参考にしながら、解決を目指したが、かなり専門的な内容になっている上に PyQGIS に関する文献が少ないのもあって、この問題を解決することはできず、手動で QGIS を用いて解析を行うことになった。「サーバで解析したデータをフロントエンド側で受信できるようにする」という問題に関しては実装を目指したのが終盤であったので、実装するための AWS のセキュリティの知識に関して学ぶ時間が足りなかった。

（※文責: 外川雄也）

5.5　メンバーごとの学習成果と自己評価

5.5.1　飯田珠羅

まず私はプロジェクトの活動目標として技術・知識の習得方法、技術・知識の応用方法、課題の解決方法を学ぶことの三つの目標を立てた。そして私は前期では主にフロントエンド側で活動を行い、後期に入ってからは時々バックエンド側の手伝いをしつつフロントエンド側で活動を行った。活動を行った成果としてフロントエンド側では前期に mac pc の数を人数分調達することが出来なかったため本を読むなどすることで swift というプログラミング言語の学習を行い、後期ではフロントエンド側であらかじめ定めた機能の実装を行った。バックエンド側ではアプリの自動化の方法や自動で実行する方法などをインターネットで探してプログラミングを行った。方法を理解し、その方法で実際にやりたいことが出来るかなどを試すことは行ったが実際に実装することは時間などの関係もあってかなわなかった。だがもう少し効率的に活動を行うことが出来れば実装することが出来たかもしれない。これらのことから私は技術・知識の習得方法と課題の解決方法の二つの目標を達成することが出来たが、技術・知識の応用方法については達成することが出来なかったので今後はより時間に気を配ることに重点を置くことを今後の目標とする。

（※文責: 飯田珠羅）

5.5.2　外川雄也

私のプロジェクトでの学習目標は「開発に必要な技術を身に付けながら、チーム開発を効率よくすすめる」であった。まず、「開発に必要な技術を身に付ける」という目標だが、これの達成度はしっかりと学習ができた技術もあれば、学習のための時間が足りず、中途半端な知識しか身につけられず、結果的に実装できなかった部分もあったので達成度はまずまずだった。私はこのチーム開発において、バックエンドを担当して、主に、サーバでの環境構築作業を行っていた。この作業に必要な技術、知識は AWS、Linux、Linux コマンド、シェル、Python などがある。私は開発を開始した当初はこれらの技術や知識ほとんどを身につけていなかった。そこで、最初は作業の中心となる AWS でのサーバの構築の仕方やサーバでの GUI の実装や必要なアプリケーション、ライブラリのインストールの仕方を学んだ。その他の技術については作業が進むにつれて必要になった時点で学ぶことにした。しかし、成果発表に近づくにつれて、作業がかなり専門的なものになっていて、なにか問題が現れても、解決するための文献などが少なく、問題解決にかなりの時間を費やしてしまった。結果的には時間的に実装できなかった機能が少なからず発生してしまった。また、もう一つの目標である「チーム開発を効率よく進める」という点はあまり達成できなかった。私の担当したバックエンドチームは作業内容が大きく別れていたが、それぞれのタスクの管理と共有がうまくできていなかったために、他の人が今どのような作業をしているか、次はどの様な作業をすればよいのかがわからなかった。チーム単位でもっと時間を費やして、タスク管理をする必要があると感じた。

（※文責: 外川雄也）

5.5.3 山田純平

私のプロジェクトでの学習目標は、技術知識の習得方法を学ぶ、技術知識の応用方法を学ぶ、作業を効率よく行う方法を学ぶ、である。私はこの目標全てを達成できた。技術知識の習得方法は成果物であるモバイルアプリケーションを作るために使用したプログラミング言語である Swift を学習するうえで学べた。技術知識の応用方法も同じくモバイルアプリケーションを作る際に学べた。作業を効率よく行う方法については、個人的に TODO リストを作り、それぞれのタスクにどのくらいの時間がかかるかをあらかじめ予想してからタスクを進めることで学んだ。この対策を行うことで、自分がどの作業に時間がかかっているか、どこに無駄があるかを知ることができた。だが、私はプロジェクトのグループリーダーとしてプロジェクトを上手く進められていなかったと思っている。今回プロジェクトが上手くいかなかった多々ある理由の中で致命的だったのがバックエンドとフロントエンド間での情報共有が不足していたこと、各プロジェクトメンバーのスキルレベルを考慮して適切な作業を分配し管理できていなかったこと、いつまでに何を達成していればいいかなどのスケジューリングが上手くできていなかったこと、そもそもスケジュール管理が上手くできていなかったことがある。これらの失敗から学び、対策を考えることで今後の自分の成長に生かしていきたい。

（※文責: 山田純平）

5.5.4 蓬畑尚輝

私のプロジェクトでの学習目標は、サーバの活用技術の会得とデータの解析によるプログラミング技術の習得、開発チームの連携を密にすることである。結果として、サーバの活用技術の会得は厳しいものとなったが、プログラミング技術の習得及びチームでの連携はできたと考える。サーバの活用技術はアプリ開発のデータを解析する部分において序盤の部分であったが、サーバの設立時点から進展が難儀なものであった。また、最終的にサーバからデータの検索ができたものの取得には至らなかったため、会得はできていないと考える。QGIS の Python によるデータ解析では、わからないことがあったらインターネットや書籍など様々な情報を利用した。また、同じ開発チームの人と相談するなどを行った。そのため、データの形式変換からデータの保存までのプログラミングが滞ることがあまりなかった。Python という言語のデータを解析する分野においては、プログラミング技術の習得ができたと考える。そして開発チームの連携では、行き詰る部分があったときやフロントエンドとバックエンドでの連携が必ず必要となるアプリに表示するためのデータ形式について話し合うことができた。

（※文責: 蓬畑尚輝）

5.5.5 菅原慎哉

私は、技術・知識の習得方法や、課題の設定・解決方法を身に着けることを目標としていた。技術・知識の習得方法について、5 月上旬から行ったリモートセンシングの学習にて QGIS の学習を行った。また、後期に参加したバックエンドの活動の際に発見した課題の解決を実現するために、Python の基礎や、QGIS に関する新たな知識について QGIS の公式サイトなどで調べたり、既に

同じ課題に直面し、解決していたメンバーからの助言をもらうことで、新たな技術・知識を学んでいった。課題の解決方法について、前期に参加したフロントエンドでの活動では、Swift での画面表示をどのようにするのかといったことや、どのような機能を追加するのかなどの課題を、自分の意見を述べることで、課題の解決に繋げることができたと考える。また、後期の活動では、先述した方法で技術・知識を新たに習得していくことで、課題の解決に取り組んだ。課題の設定方法について、今回のプロジェクトを通して解決した課題は、既に設定されていたものであったため、自分で課題を設定する方法を実践する機会を設けることができなかった。以上より、技術・知識の習得方法や、課題の解決方法を身に着けることができ、課題の設定方法を身に着けることはできなかったと考える。身に着けることができた方法は、今後の活動でも実践していき、身に着けることができなかった課題の設定方法は、今後の活動していくうえで必要となるため、適切な課題の設定を意識して取り組んでいきたい。

（※文責: 菅原慎哉）

5.5.6　山本陽平

私のプロジェクトの活動目標として、Swift の技術・知識の習得利用、技術・知識の応用、課題に対しての解決能力の向上の 3 つの目標を立てた。

私は主にフロントエンド側で活動を行った。Swift の技術の習得方法は本格的にプロジェクト学習が始まる前に、Youtube で実際に簡単なアプリを制作することである。その後プロジェクトを進めるうえで課題が発生した際、インターネットを使い、作業中に現れたエラーコード文を調査することで自分がどのようなミスをしているのか、そして同じミスが発生した場合はどのように対処すればいいのかを繰り返し確認し、技術を習得していった。そして Swift の技術の習得を進めると同時にインターネットの効率的な利用方法も分かり、改善された。後期では残っているタスクの量を考慮し、フロントエンドからバックエンドの手伝いをすることになった。よって、Swift に関する技術・知識の応用はうまく出来なかった。しかし、バックエンド側の手伝いをすることでサーバを立てる方法や、Python による画像の取得などの簡単な仕組みや技術を習得した。

よって今回立てた目標のうち、技術・知識の応用以外の 3 つ中 2 つを達成することができた。技術・知識の応用に関しては実際に新たにアプリを作ることで補うことを目標とする。

（※文責: 山本陽平）

第 6 章　コロナ

今回のプロジェクト学習の期間中は新型コロナウイルスの感染が拡大していた。新型コロナウイルスは、正式名称を COVID-19 といい、日本国内では 2020 年初頭から感染拡大し始めた。このウイルスは一般的に非常に感染しやすいものとして知られている。感染すると、発熱や咳、倦怠感、味覚の消失などの症状のほか、重篤なものだと発話障害や運動障害を引き起こす可能性がある [11]。

弊校公立はこだて未来大学を含む多くの学校は構内での感染拡大を防ぐために登校自粛処置 [12] をとった。そのため、本プロジェクトは活動日の 2/3 をオンラインで行った。

（※文責: 菅原慎哉）

6.1　オンラインでの活動の問題点と解決策

プロジェクトをオンラインで行う際に、対面で行うときと比べて、仲間の進捗を確認しづらい点、一緒に作業がしづらい点、メンバー同士の距離が縮まりづらい点が苦労する点として考えられた。これらの問題を緩和するために、問題ごとに対策をとった。

（※文責: 菅原慎哉）

6.1.1　仲間の進捗を確認しづらい点

お互いの顔を見ることがないオンラインでの活動では対面の活動と違い、コミュニケーションが生まれることが少なく、お互いの進捗が確認しづらい。この問題は 2 つの解決策によって対策した。1 つ目の対策は、活動の終了間際にお互いの進捗報告をするための時間を明示的に設けることだ。この時間で、各メンバーが予定していた作業の内容、作業の達成状況や今後の作業などについて報告する。2 つ目の対策は、メンバーがいつでも見られる場所に全員の作業計画を書くことだ。これを実現するために、Google JamBoard を利用した。Google JamBoard とは、あらゆる場所や端末で利用可能な電子ホワイトボードである。Google JamBoard の詳しい説明は第 7 章の「プロジェクト内の連絡手段」を参照してほしい。

（※文責: 菅原慎哉）

6.1.2　一緒に作業がしづらい点

オンラインでの活動では対面の活動と違い、複数人が 1 つの文書に書き込むなどの共同作業がしづらい。この問題は teams の同時編集機能を用いて解決した。teams とは、資料を共有する機能や通話をする機能などの補助機能を持ったチャットツールである。詳しい説明は第 7 章の「プロジェクト内の連絡手段」を参照してほしい。teams の同時編集機能はドキュメントはもちろん、スプレッドシートやプレゼンテーション用のスライドを複数人で同時に編集することができる。

（※文責: 菅原慎哉）

6.1.3　メンバー同士の距離が縮まりづらい点

オンラインでの活動では対面の活動と違い、コミュニケーションが生まれることが少ない。そのため、対面の活動よりもメンバー同士の距離が縮まりづらくなり、グループの連帯感が下がると考えた。この問題を解決するために、授業と授業の間の休憩時間にメンバーとちょっとしたゲームをする時間を設けた。このおかげで、メンバー同士が気軽に話せるようになった

（※文責: 菅原慎哉）

第 7 章　組織体制

7.1　プロジェクトの構成

本プロジェクトは、学生 10 人と担当教員 2 人で運営された。前期の活動はテーマを GroupA と GroupB の 2 グループに分けて活動した。また、プロジェクトリーダーを 1 名、副リーダーを 2 名選出し、そして、各グループにプロジェクトリーダーとは別にグループリーダーを 1 名ずつ選出した。プロジェクトの司会進行はプロジェクトリーダーと副リーダーが 1 回交代で回した。司会進行役が欠席の場合、次の司会進行役の人に回し、本来の司会進行役の人は次回のプロジェクトの際、司会をするようにした。グループの週報については負担が大きくならないよう順番で行うようにした。後期も同じように活動した。

（※文責: 菅原慎哉）

7.2　プロジェクト内の連絡手段

本プロジェクトにおける連絡手段は主に 4 つあった。使用した手段は以下のとおりである。

・Zoom
・Teams
・LINE
・Google Jamboard

7.2.1　Zoom について

Zoom とは複数人で同時参加可能な Web 会議アプリケーションであり、公立はこだて未来大学のオンライン授業でもたびたび使用されているツールである。このアプリケーションは PC の他にタブレットやスマートフォンでも利用が可能であり、画面の共有が可能であることから見せたいものを即座に見せることが出来るため、実際に会って会議をしているような感覚で利用が出来る。またブレイクアウトルームを用いることで話したいグループごとに分かれて活動を行うこともできるため、大人数で複数の Zoom を用いる必要がない。

（※文責: 山本陽平）

7.2.2　Teams について

Teams とは「マイクロソフト社が推奨する Microsoft365 のコミュニケーションツールで、Skype for business の後継として登場しました。Teams にはチャット・通話機能の他、ビデオ会議機能、ファイル共有機能、Office アプリとの連携機能があり、Microsoft アカウントがあれば無料での利用も可能です。」 [13] とある。本プロジェクトでは teams を用いて予定表の作成やポスターの製作、データの共有をプロジェクト内メンバーで行った。

（※文責: 山本陽平）

7.2.3 LINE について

LINE とは LINE 株式会社が運営・経営する携帯電話、スマートフォン、パソコン、タブレット、などで使用することができる無料のコミュニケーションツールである。利用者が相互に LINE をインストールすることで複数人との音声通話、チャットが利用することができる。本プロジェクトでは欠席連絡、遅刻連絡、成果物をチャットに載せることで本アプリケーションを利用した。

（※文責: 山本陽平）

7.2.4 Google Jamboard について

Google Jamboard とは上記の Zoom と同じく PC やタブレット、スマートフォンで利用可能なクラウドアプリケーションであり、文字を書くことや、写真や付箋を貼ることが出来るなどの電子ホワイトボードのように利用することが出来るツールである。本プロジェクトではデザインの構成、開発にあたってのアイディアだし、予定の確認などに利用した。

（※文責: 山本陽平）

第 8 章　制作物について

アプリケーション以外の制作物として、ポスターと発表用動画を作成した。

adobe illustrator を学校側から提供していただいたがグループのメンバーのうち adobe illustrator を使える人が一人もおらず最終的に Teams の機能である共同作業ができる PowerPoint を使用した。

（※文責: 飯田珠羅）

8.1　ポスター

ポスターは A1 サイズかつ英語と日本語のバイリンガルとあり、短く簡潔にまとめる必要があった。そこで GroupA は「背景・目的」、「アプローチ」、「結果・展望」の三本立てで書くことにした。以下の図 8.1 と図 8.2 が実際の前期と後期に作成をしたポスターである。

中間発表では「結果・展望」に現状、途中経過を書いた。そして見やすいように GroupB と GroupA を横で区切るのではなく、縦に区切ることでサブタイトルが GroupB との文字の流れの統一をした。さらにグループ毎にあったイメージカラーを選び、GroupA は青色を多く使ったポスターを制作することにした。工夫した点としては、「結果・展望」は文章ではなく箇条書きにすることによって、GroupA は何を達成したのか、どのような展望を考えているのかわかりやすく書くことにしたこと、文字の量により読み手の読みやすさを下げてしまうことを考慮し、文字数を減らし、イラストを挿入したことである。

反省・改善点としては、文字数を減らしたとあるが、ほかのプロジェクトに比べて、文字数が多く、若干読みにくい構成になってしまった点である。

（※文責: 飯田珠羅）

8.2　発表用動画

私たちのプロジェクトは Web サイトと動画を投票でどちらを作成するか決めた結果、動画を作成することにし、10 分程度で収まるように作成した。中間発表用では GroupA では PowerPoint で「GroupA の目標」、「なぜリモートセンシングを利用するのか」、「アプローチ」、「フロントエンド」、「バックエンド」、「現状」、「展望」についてのスライドを一人約 4 枚ずつとスライドに伴う音声原稿も作成した。前期の問題点としては発表動画を中間発表の間際に作成し始めたため、声の大きさやスライドの統一性をそろえることができなかった。修正や音声データを少人数で行ったため見落としや別のプロジェクトと比べて質の差が感じ取れた。

後期は中間発表用のスライドを発表用のスライドのテンプレートとし作成した。前期では途中経過を書いた部分を最終結果に変えるなどした。前期で得られた問題点をもとに後期ではスライドを作成し、音声はブラウザ上にある tool4music（M4A ファイルの音量をオンラインで調整 - M4A Volume (tools4music.info)）で音量をオンライン上で調節し、さらに、見る人が理解しやすいように文字を少なく、画像を多く使用し、アニメーションを入れた動画を作成した。著作権に関して画

図 8.1　前期の発表用ポスター

図 8.2　後期の発表用ポスター

像は著作権フリーのものを採用した。GroupB とのデザインの統一、アニメーションの統一を考慮し作成した。

反省・改善点としては後期に使用した画像の数多く、著作権について書き漏れがあり、発表直前までプロジェクトのメンバーと確認作業をした。

（※文責: 飯田珠羅）

8.3 制作物の評価

中間発表会や最終発表会の後、Google フォームにて「発表技術」「発表内容」の 2 つの項目を評価担当者の学生や教員に評価と自由記述によるコメントをいただいた。それぞれの項目は 1（非常に悪い）から 10（非常に良い）までの 10 段階評価である。

（※文責: 菅原慎哉）

8.4 前期制作物の評価

評価の数は学生と教員を含む 37 名である。発表技術についての評価の平均は 7.16（回答数 n=37、標準偏差 SD=1.85）となった。発表内容についての評価の平均は 7.32（回答数 n=37, 標準偏差 SD=1.43）となった。図 8.3 と図 8.4 は評価者より発表技術、発表内容に関する点数をグラフにまとめたものである。また自由記述でいただいたコメントの一部を抜粋する。

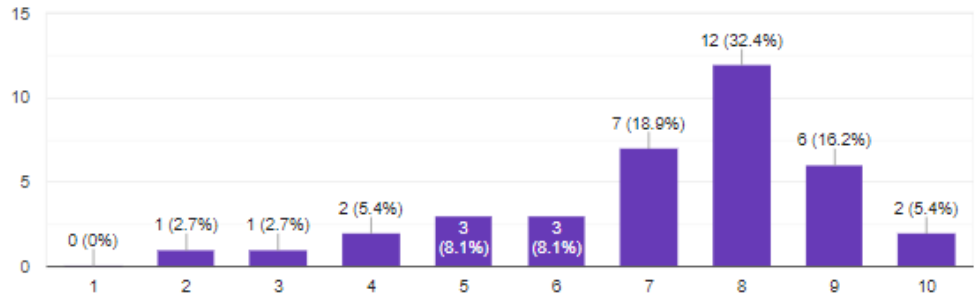

図 8.3　前期発表技術の評価

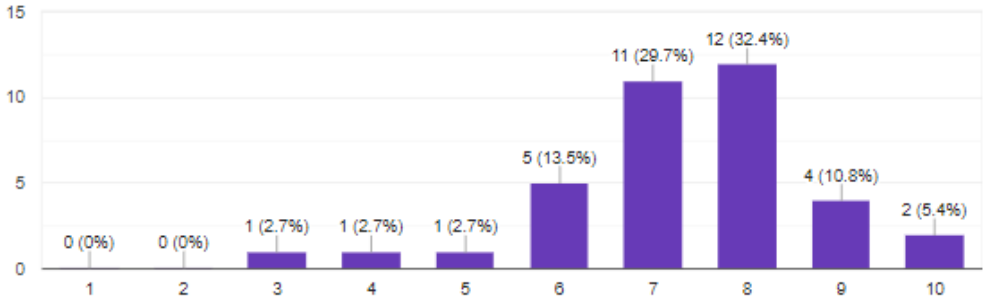

図 8.4　前期発表内容評価

8.4.1　発表技術について

良い点

・イラストなどをわかりやすく活用しており、資料全体を通して見やすかった

・聞き取りにくい言葉があまりなかった。

悪い点

・動画時間が少し長いと感じたため、もう少し短くしたりパート分けしたるすると良いと感じた

・資料について項目別にスペースが区切られておらず、詰めていることからポスターが読みにくかった。

・ポスターが文字ばかりで分かりづらい．図で示すべき．文字を使う場合は箇条書きにすべき.

・スライドは一般の人にもわかりやすいが、説明の声が明瞭でない。

・他の s 店、視野からの考えを取り入れられると完璧であると感じました。

8.4.2　発表内容について

良い点　・GroupA は目標が明確でわかりやすい。・リモートセンシングという技術についてあまり詳しくはないので良くは分かりませんが適切な目標設定ができていると思いました。・具体的な目標が設定されていて、その目標もわかりやすいものだった

悪い点　・今後の展望をより細かく記載してほしかったです。どのようなことをするのか少し曖昧でした。・とても良い目標だとは思いますが、技術的にどう解決していくのかが分からなかった。

（※文責: 菅原慎哉）

8.5　後期制作物の評価

評価の数は学生と教員を含む 35 名である。発表技術についての評価の平均は 7.03（回答数 n=35、標準偏差 SD=1.84）となった。発表内容についての評価の平均は 7.86（回答数 n=35, 標準偏差 SD=1.22）となった。図 8.5 と図 8.6 は評価者より発表技術、発表内容に関する点数をグラフにまとめたものである。また自由記述でいただいたコメントの一部を抜粋した。

8.5.1　発表技術について

良い点

・動画が見やすくて良いと思った。

・動画に動きがついていてとても見やすかったです。

・図を使うなどして、見やすいスライドだった

悪い点

・内部で何が起こっているのか、今一つ理解することができなかった。

・分かりやすく説明されていたが、所々ノイズが酷かったため。

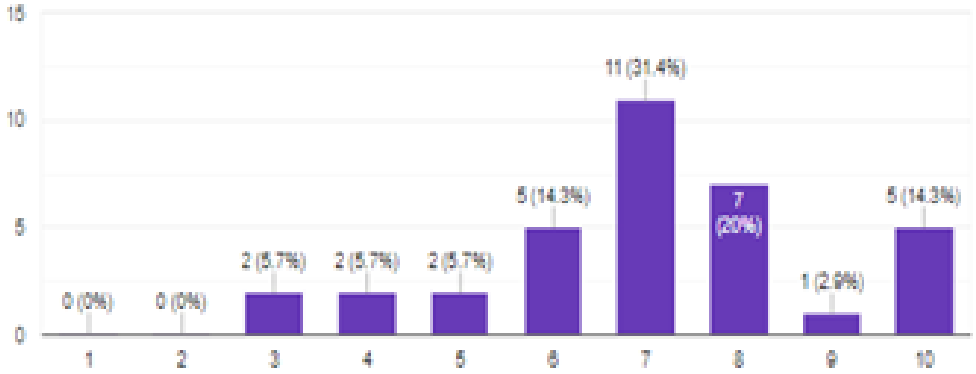

図 8.5　後期発表技術の評価

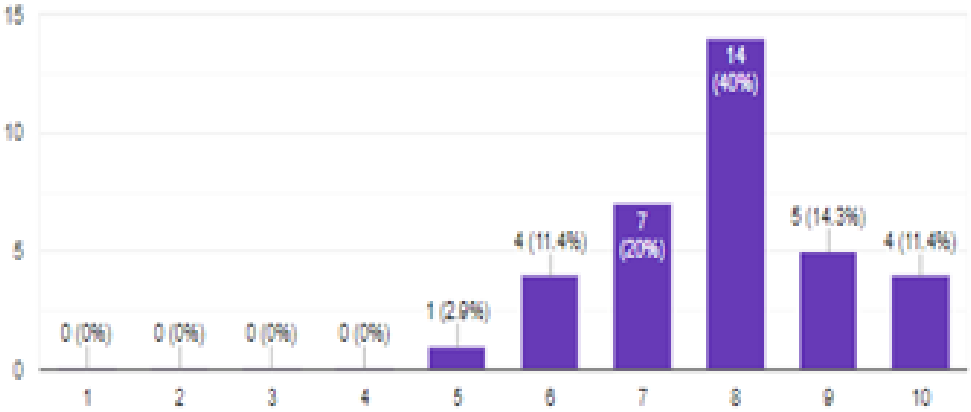

図 8.6　後期発表内容の評価

・説明がいきなりフロントエンド、バックエンドという話から始まっているのですが、もう少し技術的な背景をもたないひとにやさしい説明ができるとよいかと思いました。

8.5.2　発表内容について

良い点

・プロジェクトの目標設定や計画に関して、特に問題なかったように思う

・海のデータを解析し、魚の生息場所を予測できるというのはとても面白いと思います。

・とてもわかりやすかったです。

悪い点

・出来ていないことを書いている部分が多いと感じた。

・既存のものとの差別化が気になった。

・動画を見ても、それぞれの想定する利用者と利用方法、利用するとどのようなことが得られるのかが分かりづらい。

・もう少し詳しく動画で説明してほしい部分がありました

（※文責: 外川雄也）

第 9 章　プログラムについて

9.1　フロントエンド

ここでは、フロントエンド側を作るにあたって使用した言語と開発ツールの紹介、アプリを作るために作成したプログラムについての解説を行う。

フロントエンドは Swift という言語で作成した。 Swift とは 2014 年に Apple 社が発表した iOS、Mac、AppleTV、AppleWatch などの Apple 製品向けのアプリを開発するためのオープンソースのプログラミング言語である。

9.1.1　開発ツールについて

Xcode

Swift でフロントエンドを開発するにあたり、Xcode というツールを使用した。Xcode とは、Apple 社が開発しているアプリケーションの統合開発環境である。統合開発環境とは、英語で Integrated Development Environmet と書き、開発に必要な道具が完全にそろえられた環境を意味する。その名の通り、コーディングやコンパイルなどの基本的な機能に加えて、コードの検索機能やコードの補完機能など様々な便利機能が使える。

9.1.2　プログラムファイルについて

ここではアプリケーションを作成する際に使用するプログラムファイルの一部について解説する。

（※文責: 山田純平）

AppDelegate.swift

アプリを作った段階で作られているプログラムファイルである。アプリが開かれたタイミング、閉じられたタイミングで呼ばれる関数が記述されている。今回このファイルは変更を加えていないので詳しい解説は省く。

SceneDelegate.swift

SceneDelegate.swift は AppDelegate.swift と同じく、アプリを作った段階で作られているプログラムファイルである。今回、このファイルは変更を加えていないので詳しい解説は省く。

ViewController.swift

ViewController.swift はアプリが開かれて一番最初に表示されるビューコントローラーに関するプログラムを記述するファイルである。ビューコントローラーとは、1 つの画面に表示される UI 部品の表示、配置、アニメーションなどを管理する UI 部品である。

mapView の宣言と初期化

リスト 9.1　ViewContoroller.swift(1)

```
@IBOutlet weak var mapView: MKMapView! {
        didSet {
            mapView.delegate = self
            let displayObjects = parseGeoJSON()
            mapView.addOverlays(displayObjects)
            mapView.showsUserLocation = true
            mapView.addAnnotations(fisherShopPointAnnotation)
        }
}
```

mapView とは何か

mapView は画面に表示されるマップに関する操作などを管理するための変数である。変数の先頭につけられた@IBOutlet によって InterfaceBuilder 上の mapView の UI に関連づけられる。 InterfaceBuilder とは、画面への部品をプログラムを書くことなく画面に配置できるツールである。

didSet について、なぜこれを使用したか

didSet は変数に変更があった時、変更後に 内の処理を行える。この特性を利用して、mapView に InterfaceBuilder 上の MapView のインスタンスが代入されたときに初期化の処理として 内の処理を行わせている。didSet を使用することで、mapView の初期化処理を一箇所にまとめ、コードを見やすくしている。

delegate について

delegate は他のクラスに自分の処理を移譲する仕組みである。mapView.delegate に self(ここでは ViewController クラス) を代入することによって、ViewController クラスが mapView に関する処理を管理することになる。

mapView.addOverlays(displayObjects) について

ParseGeoJSON() 関数で漁場を示している JSON ファイルを MKOverlay プロトコルに準拠したオブジェクト群に変換している。そのオブジェクト群を displayObjects に代入し、mapView.addOverlays 関数を使い、マップに覆いかぶせている。ParseGeoJSON() 関数の内容に関しては下記で詳しく説明する。

mapView.showsUserLocation について

MapView.showsUserLocation は mapView がユーザーの位置を表示しようとするかどうかを表す Bool 変数である。

mapView.addAnnotations() について

MapView.addAnnotations(fisherShopPointAnnotation) は mapView に釣り具屋のアノテーションを追加している。アノテーションとは、mapView に表示されるピンのことである。

locationManager の宣言と初期化

リスト 9.2　ViewContoroller.swift(2)

```
private lazy var locationManager: CLLocationManager = { [weak self] in
    let manager = CLLocationManager()
    //selfを参照できない（メモリの循環参照問題）？？
    manager.delegate = self
    manager.desiredAccuracy = kCLLocationAccuracyThreeKilometers
    manager.requestWhenInUseAuthorization()
```

```
7       manager.requestLocation()
8       return manager()
9   }
```

locationManager とは何か

LocationManager は位置情報を取得するための変数である。locationManager は CLLocationManager クラスの変数である。CLLocationManager とは、ユーザーの位置情報を取得と取得の際に設定するパラメータさまざまなプロパティを管理するクラスである。

クロージャについて

LocationManager 変数はクロージャ—の返り値を代入することによって初期化している。クロージャ—とは、わかりやすく言うと で囲んだ中にある処理を実行する即席の関数である。mapView の didSet と同じように、locationManager に関する処理をまとめて記述することでコードの可読性を向上させようとした。

lazy とは何か

変数の前に lazy 修飾子をつけると、変数が使用されるまで初期化処理が行われなくなる。今回はその性質を利用して、ViewController インスタンス (self) が生成される前に、self を参照する locationManager 初期化処理が行われないようにしている。

weak self とは何か、循環参照について

[weak self] は self と locationManager を初期化する際に使用するクロージャの循環参照を解消するために記述している。循環参照とは、複数のインスタンスがお互いを参照しあってループを成している状態のことを言う。Swift は ARC というシステムでオブジェクトを管理している。他のオブジェクトから参照されているオブジェクトはメモリに残り、他のオブジェクトから参照されていないオブジェクトはメモリから破棄されるという仕組みである。なので、循環参照が起こっているオブジェクトはそれを解消しない限り、メモリから永遠に破棄されなくなってしまう。メモリから破棄されないオブジェクトが存在すると、メモリの空き領域が減少し、実行中のプログラムが必要なメモリ領域を確保できず停止・異常終了することがあり、非常に危険である。循環参照を回避するために、[weak self] を記述しクロージャから self への参照を弱参照にした。弱参照とは、特殊なオブジェクト参照のやり方である。弱参照でオブジェクトを参照しても、他のオブジェクトから参照されていない限りメモリから破棄される。

manager.desiredAccuracy について

manager.desiredAccuracy は位置情報を取得する際、どのくらいの精度で取得するかを表す変数である。今回指定している kCLLocationAccuracyThreeKilometers は 3 キロメートル以内の精度を意味している。

manager.requestWhenInUseAuthorization()

manager.requestWhenInUseAuthorization() はユーザーにアプリでの位置情報の使用の許可を求めるための関数である。

manager.requestLocation()

manager.requestLocation() は一度だけ位置情報を取得する関数である。

longPressMap(_ sender: UILongPressGestureRecognizer) の処理

リスト 9.3　ViewController.swift(3)

```
1   @IBAction func longPressMap(_ sender: UILongPressGestureRecognizer) {
```

```
2         let location: CGPoint = sender.location(in: mapView)
3         if sender.state == UIGestureRecognizer.State.ended {
4             locationManager.requestLocation()
5
6             let mapPoint: CLLocationCoordinate2D = mapView.convert(location
7 , toCoordinateFrom: mapView)
8
9             let annotation = MKPointAnnotation()
10            annotation.coordinate = CLLocationCoordinate2DMake(mapPoint.latitude
11 , mapPoint.longitude)
12            mapView.addAnnotation(annotation)
13        }
14 }
```

longPressMap とは何か

longPressMap は mapView を長押ししたときに実行される関数である。引数は UILongPressGestureRecognizer クラスが指定されている。このクラスは長押しした場所や長押しの始まり、終わりのタイミングであるかなどの長押しの情報を管理しているクラスである。今回の処理ではこのクラスの引数を用い、長押しの終わりのタイミングの時、長押しされた箇所にアノテーションを立てる処理を記述している。

parseGeoJSON 関数の処理

リスト 9.4　ViewContoroller(4)

```
1    func parseGeoJSON() -> [MKOverlay] {
2         //1 URL取得
3         guard let url = Bundle.main.url(forResource: "1113take1",
4  withExtension: "json") else {
5             fatalError("Unable to get geojson")
6         }
7         //2 NSGeoJSONObjectにデコード
8         var geoJson = [MKGeoJSONObject]()
9         do {
10            let data = try Data(contentsOf: url)
11            geoJson = try MKGeoJSONDecoder().decode(data)
12        } catch {
13            fatalError("Unable to decode geojson")
14        }
15        //MKGeoJSONFeature：
16        //MKPolygon
17        //geoJSONObject(プロトコル)をMKGeoJSONFeature(クラス)に変
18        var overlays = [MKOverlay]()
19
20       //3
21        for item in geoJson {
22            if let feature = item as? MKGeoJSONFeature {
23                for geo in feature.geometry {
24                    if let polygon = geo as? MKPolygon {
25                        overlays.append(polygon)
26                    } else if let point = geo as? MKPointAnnotation {
27                        overlays.append(MKCircle(center: point.coordinate,
28  radius: 1000))
29                        print(point)
30                        print(overlays)
31                    }
32                }
33            }
```

ParseGeoJSON 関数は漁場が示された JSON ファイルを MKOverlay プロトコルに準拠しているオブジェクトの配列に変換する関数である。MKOverlay とは、地図に上書き描画するための図形のオブジェクトが満たすべきプロトコルである。プロトコルとは、他の言語でいうところのインターフェースとほぼ同一であり、プロトコルを満たしているオブジェクトはプロトコルに準拠しているという。1 では JSON ファイルの URL を取得している。今回は JSON ファイルを提供するサーバが完成しなかった。そのため、代わりにローカルのテストデータを取得している。guard は if と同じように条件分岐を行うための文法である。if は条件式が true の時に処理を行うのに対して、guard は条件式が false の時に処理を行う。guard let ... は guard 文の応用であり、url に正常に値が代入できなかった時、処理が行われる。ここでは fatalError を出す処理が行われている。2 では JSON ファイルを MKGeoJSONObject クラスの配列に変換している。3 では MKGeoJSONObject クラスを MKOverlay プロトコルに準拠しているオブジェクトの配列に変換している。

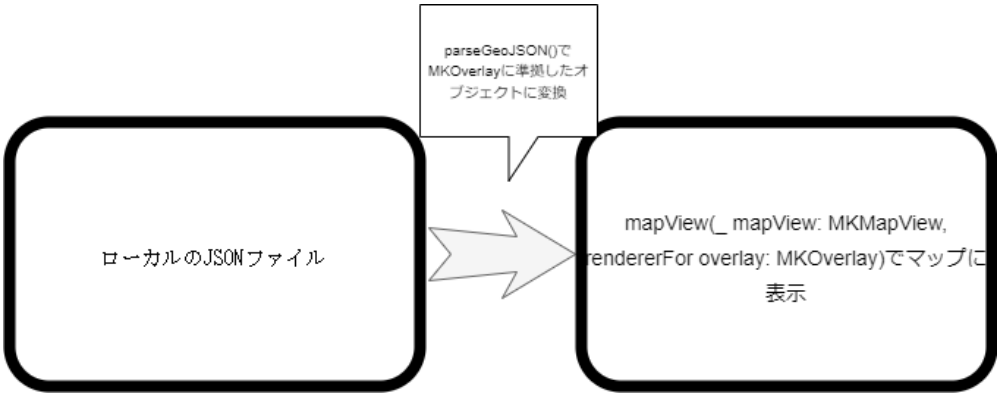

図 9.1　parseGeoJSON の流れ

mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) の処理

リスト 9.5　ViewController.swift(5)

```
1  func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) ->
2   MKAnnotationView? {
3
4          guard !(annotation is MKUserLocation) else {
5              return ni
6          }
7
8          if annotation.subtitle == "釣り具" {
9              let annotationView = MKPinAnnotationView(annotation: annotation,
10  reuseIdentifier: nil)
11             annotationView.pinTintColor = UIColor.blue
12             annotationView.canShowCallout = true
13             return annotationView
14         }
15
16         let customAnnotationView = CustomAnnotationView(annotation: annotation
17 , reuseIdentifier: nil)
18         customAnnotationView.getWeatherData(latitude: annotation.coordinate
19 .latitude, longitude: annotation.coordinate.longitude)
20         return customAnnotationView
21     }
```

mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) は mapView の annotation に annotationView を設定する関数である。

ここでは、現在位置を表すアノテーションには annotationView を設定せず、釣り具屋のアノテーションには釣り具屋名を表示する annotationView を設定し、それ以外のアノテーションには天気を表示できるようにしている。

mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) の処理

リスト 9.6　ViewController.swift(6)

```
1  func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) ->
2   MKOverlayRenderer {
3          if let polygon = overlay as? MKPolygon {
4              let renderer = MKPolygonRenderer(polygon: polygon)
5              renderer.fillColor = UIColor.blue
6              renderer.strokeColor = UIColor.black
7              return renderer
8          } else if let circle = overlay as? MKCircle {
9              let renderer = MKCircleRenderer(circle: circle)
10             renderer.fillColor = UIColor.blue
11             renderer.strokeColor = UIColor.black
12             return renderer
13         } else if let polyline = overlay as? MKPolyline {
14             let renderer = MKPolylineRenderer(polyline: polyline)
15             renderer.strokeColor = UIColor.blue
16             renderer.lineWidth = 2.0
17             return renderer
18         }
19         return MKOverlayRenderer(overlay: overlay)
20
21     }
```

mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) は mapView に追加された MKOverlay プロトコルに準拠したオブジェクトごとに呼び出される関数である。ここではプロトコルに準拠したオブジェクトが MKPolygon クラスである時、MKCircle クラスである時、MKPolyline クラスである時の 3 パターンに処理を分けて書いている。MKPolygon クラスは漁場の位置を示している MKOverlay プロトコルに準拠したオブジェクトを描画するための処理を書いている。MKPolyline クラスは経路を表示するための処理を書いている。

mapView(_ mapView: MKMapView, didSelect view: MKAnnotationView) の処理

リスト 9.7　ViewContoroller.swift(7)

```
1  func mapView(_ mapView: MKMapView, didSelect view: MKAnnotationView) {
2
3          let presentLocation = nowLocation
4          if let annotation = view.annotation {
5              mapView.getRoute(source: nowLocation, dest: annotation.coordinate)
6          }
7      }
```

mapView(_ mapView: MKMapView, didSelect view: MKAnnotationView) はマップ上のピンを

タップするなどして、ピンが選択状態になったときに実行される関数である。ここでは、getRoute 関数を用いて、現在地からピンの場所までのルートを検索して表示する処理を行っている。

mapView(_ mapView: MKMapView, didDeselect view: MKAnnotationView) の処理

リスト 9.8 ViewContoroller.swift(8)

```
1  func mapView(_ mapView: MKMapView, didDeselect view: MKAnnotationView) {
2          for overlay in mapView.overlays {
3              if overlay as? MKPolyline != nil{
4                  mapView.removeOverlay(overlay)
5              }
6          }
7      }
```

mapView(_ mapView: MKMapView, didDeselect view: MKAnnotationView) はマップ上のピンの選択状態が解除されたときに実行される関数である。ここでは、ピンが選択状態になった時に表示されたルートを消す処理を行っている。

locationManager

リスト 9.9 ViewContoroller.swift(9)

```
1  func locationManager(_ manager: CLLocationManager,
2   didUpdateLocations locations: [CLLocation]) {
3              let location = locations.first
4              let latitude = location?.coordinate.latitude
5              let longitude = location?.coordinate.longitude
6              self.latitudeNow = String(latitude!)
7              self.longitudeNow = String(longitude!)
8          }
```

MKMapView+Extention.swift

MKMapView+Extention クラスは MKMapView の Extention を定義しているファイルである。

リスト 9.10 ViewContoroller.swift(10)

```
1  func getRoute(source: CLLocationCoordinate2D, dest: CLLocationCoordinate2D) {
2          let sourcePlaceMark = MKPlacemark(coordinate: source)
3          let destPlaceMark = MKPlacemark(coordinate: dest)
4          //1 経路探索のリクエストを編集
5          let directionRequest = MKDirections.Request()
6          directionRequest.source = MKMapItem(placemark: sourcePlaceMark)
7          directionRequest.destination = MKMapItem(placemark: destPlaceMark)
8          directionRequest.transportType = .automobile
9          //2
10         let directions = MKDirections(request: directionRequest)
11         directions.calculate { (response, error) in
12             guard let directionResponse = response else {
13                 if let error = error {
14                     print("we have error getting directions==
```

```
15 \(error.localizedDescription)")
16                 }
17                 return
18             }
19             let route = directionResponse.routes[0]
20             self.addOverlay(route.polyline, level: .aboveRoads)
21         }
22     }
```

getRoute(source: CLLocationCoordinate2D, dest: CLLocationCoordinate2D) は 2 次元座標を 2 つ入力することで、その 2 か所をつなぐ最短経路を表す MKPolyline オブジェクトを MKMapView クラスに加える。1 でリクエストオブジェクトを生成、編集し、2 でリクエストを送り、経路を表す MKPolyline オブジェクトを加えている。

CustomAnnotationView.swift CustomAnnotationView クラスが記述されている。CustomAnnotationView クラスは、マップを長押しすることで生成したピンをタップしたときに出てくる吹き出しに関するクラスである。

weatherData

リスト 9.11 ViewContoroller.swift(11)

```
1  var weatherData: OpenWeatherData? {
2          didSet{
3              canShowCallout = true
4              if weatherData != nil {
5                  print("didSet時",weatherData?.weather[0].icon)
6                  let rightImageView = UIImageView(image: getWeatherImage(icon:
7   weatherData!.weather[0].icon ))
8                  rightImageView.frame = CGRect(x: 0, y: 0, width: 30, height: 30)
9                  rightImageView.contentMode = .scaleAspectFit
10                 rightImageView.backgroundColor = UIColor.lightGray
11                 detailCalloutAccessoryView = rightImageView
12             }
13
14         }
15     }
```

getWetherData

リスト 9.12 ViewContoroller.swift(12)

```
1  func getWeatherData(latitude: CLLocationDegrees, longitude: CLLocationDegrees) {
2
3          let url: URL = URL(string: "http://api.openweathermap.org/data/2.5/
4  weather?lat=\(latitude)&lon=\(longitude)&
5  APPID=97e21808b7b889be6e8f5f656ab3e361")!
6
7          let task: URLSessionTask = URLSession.shared.dataTask(with: url,
8   completionHandler: {data, response, error in
9              do {
10                 let decoder: JSONDecoder = JSONDecoder()
11                 let decodedWeatherData = try decoder.decode(OpenWeatherData.self,
12  from: data!)
13                 DispatchQueue.main.async {
14                     self.weatherData = decodedWeatherData
15                 }
```

```
16                  print("オープンデータ", self.weatherData)
17
18              }
19              catch {
20                  print(error)
21              }
22          })
23          task.resume()
24      }
```

getWeatherData(latitude: CLLocationDegrees, longitude: CLLocationDegrees) 関数は OpenWeatherMap の API に緯度と経度の情報を送り、その場所の天気のデータを受け取り、weatherData 変数に格納する。

リスト 9.13　ViewContoroller.swift(13)

```
1  func getWeatherImage(icon: String) -> UIImage {
2          let fixIcon = icon.dropLast() + "d"
3          let url = URL(string: "http://openweathermap.org/img/wn/\(fixIcon).png")!
4
5          let data = try? Data(contentsOf: url)
6          let image = UIImage(data: data!)
7
8          return image!
9      }
```

OpenWeatherData

リスト 9.14　ViewContoroller.swift(14)

```
1  struct OpenWeatherData: Decodable {
2      struct weather: Decodable {
3          let id: Int
4          let main: String
5          let description: String
6          let icon: String
7      }
8      let weather: [weather]
9  }
```

（※文責: 山田純平）

9.2　バックエンド

バックエンド側を作るにあたって使用した言語と開発ツールの紹介、アプリを作るために作成したプログラムについての解説を行う。

9.2.1　言語について

Python の説明

Python は Web アプリケーションや組み込み開発など様々な開発において使用される言語である。人工知能の研究や機械学習をさせる上で Python を利用するケースが多い。Ruby

や JavaScript と同じく動的型付け言語であり、変数や関数の引数には型がしていされていない。

Python を選択した理由、Python の利点

バックエンドは Python という言語で作成した。サーバ処理アルゴリズムの作成や、QGIS での解析アルゴリズムの作成に Python を利用した理由は 3 つある。

一つ目は Python には豊富なライブラリがあり、プログラムを作成する上で利用しやすい点が挙げられる。

二つ目は、参考資料が書籍やインターネットに多く存在しているため、学習コストが低いからである。

三つ目は、今回の処理アルゴリズムで文字を入力することはないと考え、動的型付け言語でプログラミングしたほう開発しやすいと考えたためである。

9.2.2 開発ツールについて

QGIS-python コンソール

Python でフロントエンドを開発するにあたり、QGIS の python コンソールを使用した。QGIS はプラグインアーキテクチャで設計されており、Python コンソールを利用することで QGIS に備わっている PythonAPI が利用しやすい点が挙げられる。Python コンソールとは QGIS での Python コマンド実行用の対話型シェルであり、Python スクリプトを手動で編集して保存できるファイルエディタも存在する。Python コンソールとエディタはどちらも PyQSQScintilla2 パッケージに基づいている。

9.2.3 プログラムファイルについて

ここではアプリケーションを作成する際に使用するプログラムファイルについて解説する。

startup.py

Startup.py はアプリが開かれて一番最初に実行される Python ファイルである。QGIS を起動するたびに、ユーザーの Python のホームディレクトリの中で”:file: ‘startup.py’ ”という名前のファイルが検索され、そのファイルが存在する場合埋め込まれた Python インタプリタによって実行される。解析アルゴリズムはすべてこのファイル内で順に実行され、最後の処理が終了すると同時に QGIS が終了する。

Test2.py の解説

G-portal からファイルを取得するための P ython ファイルである。一定時間ごとに実行され、その時点での最新の h5 ファイルを G-portal から取得する。

G-portal サーバへの接続

リスト 9.15　Test2.py(1)

```
1 # SFTP接続先の設定
2 HOST = 'ftp.gportal.jaxa.jp'
3 PORT = 2051
4 SFTP_USER = 'User'
5 SFTP_PASSWORD = 'Password'
```

まず G-portal のサーバに接続する。HOST と PORT には G-portal の URL を使用し、SFTP 接続を行うユーザーとユーザーのパスワードを入力する。

paramiko による SSH 接続

リスト 9.16　Test2.py(2)

```
downloadProductPaths = []
client = paramiko.SSHClient()
client.set_missing_host_key_policy(paramiko.AutoAddPolicy)
client.connect(HOST, port=PORT, username=SFTP_USER, password=SFTP_PASSWORD)
```

Paramiko.SSHClient() では知らないホストに接続する際の方針を決定する。set_missing_host_key_policy は接続しようとしたサーバーホストキーがなかった場合にどうするのかを設定している。接続先のサーバが不明な場合、ホストキーが自動で追加されるようになっている。その後、SSH サーバに接続し、認証する。

Raster.py の解説

リスト 9.17　Raster.py(1)

```
startTime = "2021-01-01T00:00:00Z"
endTime = "2021-01-01T03:00:00Z"
command = "curl -s " + "\'https://gportal.jaxa.jp/csw/csw?service=CSW&version
=3.0.0&request=GetRecords&outputFormat=application/json&count
=3000&datasetId=10002002&startTime=" + startTime + "&endTime=" + endTime +
 "&orbitDirection=Descending&sceneNumber=10&resolution=1km\'" + " | jq
\'.features[].properties.product.fileName\'"
try:

    res = subprocess.check_output(command,shell=True,encoding='utf-8')
    files = res.split()
    for e in files:
        fileSplited = e.split("/")
        productName = fileSplited[12]
        pathNum = int(productName[20] + productName[21] + productName[22])
#パス番号、ppp
        if 25 <= pathNum and pathNum <= 75:
            #例　/standard/GCOM-C/GCOM-C.SGLI/L2.OCEAN.SST_/2/2021/01/01/
GC1SG1_202101010140J05510_L2SG_SSTDK_2000.h5
            downloadProductPaths.append("/" + "/".join(fileSplited[4:])[:-1])
except Exception as e:
    print('=== エラー内容 ===')
    print(e)

try:
    sftp_connection = client.open_sftp()
    for e in downloadProductPaths:
    #ファイル名のみではなく、ディレクトリも指定
        sftp_connection.get(e , '/Users/Username/Desktop/download-4.h5')
except Exception as e:
    print('=== エラー内容 ===')
    print(e)
```

データの取得には Subprocess コマンドを使用する。Subprocess は Python のプログラムからコマンドプロンプトや Shell 上で実行処理ができたり、アプリの起動ができる Python の標準ライブラリである。Subprocess.check_output() は Subprocess ライブラリ内の関数の一つであり、引

数でコマンドを実行し、その出力を返す。検索には split() を使用した。split() は文字列をある単語を基準として区切った単語のリストを返す。今回は”/”をもとに分割したリストを取得した。そして、取得するファイルがあった場合は append() を使用し取得するためのパスに代入した。append() は Python のリスト型メソッドの一つであり、リストの末尾に要素を一つ追加する。その際 join() を使用して検索したファイルのパスを代入できるようにする。join() は Python の str メソッドの一つであり、str オブジェクトオブジェクトを結合するためのメソッドである。それらを踏まえて try で実行する。

Import processing の解説

以降は QGIS で行う処理であり、その中で import processing を行わなければ実行できないものがある。processing はいろいろなプロバイダから提供されている空間データ加工のアルゴリズム群を組み合わせて利用できるフレームワークである。今回 processing を利用した理由として、QGIS の解析するための処理アルゴリズムが豊富であることと、複数のデータ加工アルゴリズムを使うことで解析が容易になることから利用することとした。processing を使用したコードの部分は processing.run() から表される。

gdal:cliprasterbyextent によるマップの切り抜き処理

リスト 9.18　Raster.py(2)

```
filePath='G:file1.tif'#GeoTIFFにしたファイルのパス
rlayer = QgsRasterLayer(filePath, 'file1')
iface.addRasterLayer(filePath,'file1')

processing.run("gdal:cliprasterbyextent", {
                'INPUT':rlayer,#GeoTIFFにしたファイルのパス
                'PROJWIN': '139.543464467,141.893972081,40.781281726,42.124428934',
#切り取り範囲
                'DATA_TIPE': 0,
               'OUTPUT': 'G:/clip.tif'})#出力ファイル


filePath='G:clip.tif'#切り取った出力ファイルのパス
```

gdal:cliprasterbyextent はレイヤを GDAL クリップラスターによって提供される QGIS アルゴリズムである。今回の処理では、入力レイヤの函館周辺におけるクリッピングを行った。”PROJWIN”で出力するラスターの範囲を指定する。取得するファイル毎の地点は一定であるため、範囲の変更は行わない。”DATA_TYPE”では出力するファイル形式に変化がないため、入力ファイルのファイル形式のままで出力できるように”0”を入力した。”OUTPUT”では出力ファイルの名前を入力した。

QgsRasterCalculatorEntry による計算前処理

リスト 9.19　Raster.py(3)

```
entries = []
# Define band1
test1 = QgsRasterCalculatorEntry()#計算できるようにするおまじない
test1.ref = 'test@1'
test1.raster = testLayer
test1.bandNumber = 1
entries.append( test1 )
```

図 9.2　解析前の画像

図 9.3　切り抜きを行った後の画像

QgsRasterCalculatorEntry は計算するラスタレイヤの属性やバンド番号を表す関数である。次に記述する QgsRasterCalculator で処理する上でラスタレイヤのリストを渡す必要があるため事前にレイヤ属性とバンド番号を取得する。

QgsRasterCalculator によるレイヤの再計算

リスト 9.20　Raster.py(4)

```
1 calc = QgsRasterCalculator( 'test@1*0.0012-10',
2                             'G:/abc.tif', #計算後の出力ファイルのパス
3                             'GTiff',
4                             testLayer.extent(),
5                             testLayer.width(),
6                             testLayer.height(),
7                             entries )
8 calc.processCalculation()#計算処理
```

計算には QgsRasterCalculator を利用した。このクラスのパラメータに'EXTENT'があり、出

力範囲が指定できる。CRS は outputsCrs パラメーターで指定される。しかし、前段階でレイヤの範囲をクリッピングしているため、extent(),width(),height() を利用してクリッピングしたあとのレイヤ全体を計算を行う。

qgis:slope によるレイヤ全体の微分

リスト 9.21　Raster.py(5)

```
processing.run("qgis:slope",{#傾斜を調べる処理
                'INPUT':'G:abc.tif',#計算後の出力ファイルのパス
                'Z_FACTOR': 1,
               'OUTPUT': 'G:abc2.tif'})#傾斜計算後の出力ファイルのパス
iface.addRasterLayer('G:abc2.tif')
```

微分では processing から qgis:slope を利用する。'Z_FACTOR' は垂直方向への誇張である。結果を出す際により差異が激しいところが見えやすくなるようにする属性であるが、今回はあまり海面温度の細分化を行う必要がなかったため、'Z_FACTOR' の値を 1 とする。

QgsSingleBandPseudoColorRenderer によるレイヤの色分け

リスト 9.22　Raster(6)

```
1
layer = iface.activeLayer()#現時点で選択しているレイヤー(この場合はabc2)
renderer = layer.renderer()
layer.renderer()#レイヤーの型番？
provider = layer.dataProvider()
layer.renderer().type()#レイヤーのバンドタイプ
ver = provider.hasStatistics(1, QgsRasterBandStats.All)
stats = provider.bandStatistics(1, QgsRasterBandStats.All)

if ver is not False:
    print ("minimumValue = ", stats.minimumValue+89.7)#最小値
    print ("maximumValue = ", stats.maximumValue)#最大値

if (stats.minimumValue< 0):#なくする
   min = NULL
else:
    min= stats.minimumValue+89.7

max = stats.maximumValue

range = max - min    #範囲
add = range//2
interval = min + add     #配列の大きさぶん
2
colDic = {'red1':'#fff5f0',
          'red2':'#fc9272',
          'red3':'#67000d',}
valueList =[min, interval, max]
lst = [QgsColorRampShader.ColorRampItem(valueList[0], QColor(colDic['red1'])),
#白に近い赤
       QgsColorRampShader.ColorRampItem(valueList[1], QColor(colDic['red2'])),
#白と赤の中間
       QgsColorRampShader.ColorRampItem(valueList[2], QColor(colDic['red3']))
#赤に近い赤
       ]
3
```

```
37 myRasterShader = QgsRasterShader()#ラスターがシェーダー機能を使えるようにする
38 myColorRamp = QgsColorRampShader()#カラーランプでシェーダ機能を使えるようにする
39 myColorRamp.setColorRampItemList(lst)
40 myColorRamp.setClip(True)#カラーランプで絞れない範囲は非表示
41 myColorRamp.setColorRampType(QgsColorRampShader.Interpolated)#線形的になるように
42 分類
43 4
44 myRasterShader.setRasterShaderFunction(myColorRamp)
45 myPseudoRenderer = QgsSingleBandPseudoColorRenderer(layer.dataProvider(),
46                                                      layer.type(),
47                                                      myRasterShader)
48 layer.setRenderer(myPseudoRenderer)#カラーランプによるシェーダの実行
49 layer.triggerRepaint()#レイヤ更新
```

QgsSingleBandPseudoColorRenderer は QgsRasterRenderer クラスに基づいた疑似バンドカラー用のラスターレンダラークラスである。パラメータとして、順に QgsRasterInterface による input,band,shader を設定する。

まず、1 ではその時点でアクティブ状態になっているレイヤを選択し、レイヤのプロパイダを取得した。その後、バンド統計を取得し、色分けを行う際の最小値と最大値を設定した。色分けに使用する値以外を無視できるようにするため範囲外の値に NULL を代入した。2 では vakueList に色分けを行う範囲を代入し、色分けを行うための lst では QgsColorRampShader.ColorRampItem を利用し、色を変化させる閾値を入力する。 QgsColorRampShader.ColorRampItem はベースが sip.warpper のデフォルトのコンストラクタで、カスタムカラーマップを返す（3)。

QgsRasterShader() は QgsRasterShader クラスのコンストラクタでラスタがシェーダ機能を利用できるようにする。また、QgsColorRampShader() は QgsColorRampShader クラスのコンストラクタであり、カラーランプによるシェーダ機能を利用できるようにする。これによりランプ内の値の範囲内のリストに基づいて色付けが行われるようになる。setColorRampItemList() では実行した際のカラーマップを設定する。

SetClip() では QgsColorRampShader クラスの関数で、シェーダが値を範囲外にレンダリングするかどうかを設定する。引数は bool である。今回は使用する値以外を無視するため True で行う。setColorRampType() は qgscolorrampsheder.cpp という QgsRasterLayer クラスであり、カラーランプよる色の分け方を設定する。4 では、

SetRasterShaderFunction() が QgsRasterShederFunction 関数であり、シェーダ関数を設定できるようにするパブリックメソッドである。ラスタシェーダがシェーダ関数のインスタンスを取得する。

最後に QgsSingleBandPseudoColorRenderer() で該当するパラメータを入力し、SetRenderer() で実行する。

RastertoVector.py について

RastertoVector.py ではラスタデータのベクタ化を行う。

gdal:polygonize によるラスタのベクタ化

リスト 9.23　RastertoVector.py

```
1 from qgis import processing
2 iface.addRasterLayer('/testdeta.tif','testdata')
3 processing.run ( "gdal:polygonize", { 'BAND' : 1, 'EIGHT_CONNECTEDNESS' : False,
4  'EXTRA' : '', 'FIELD' : 'DN', 'INPUT' : '/testdeta.tif', 'OUTPUT' :
5  '/testdeta.gpkg' } )
6 filePath = '/testdeta.gpkg'
```

図 9.4　色分けを行ったデータ

```
7  iface.addVectorLayer(filePath,'','ogr')
8  processing.run("native:savefeatures", {'INPUT':'/testdeta', 'OUTPUT':
9  '/OUTPUT.geojson', 'LAYER_NAME':'output', 'DATASOURCE_OPTIONS':'',
10  'LAYER_OPTIONS':''})
```

RastertoVector.py では processing から gdal:polygonize を利用した。gdal:polygonize は共通のピクセル値を共有するラスター内のピクセルの接続されたすべての領域のベクトルポリゴンを作成する。その後、 processing から native:savefeatures を利用した。 native:savefeatures は GDAL ライブラリの一つであり、選択した地物を保存する。

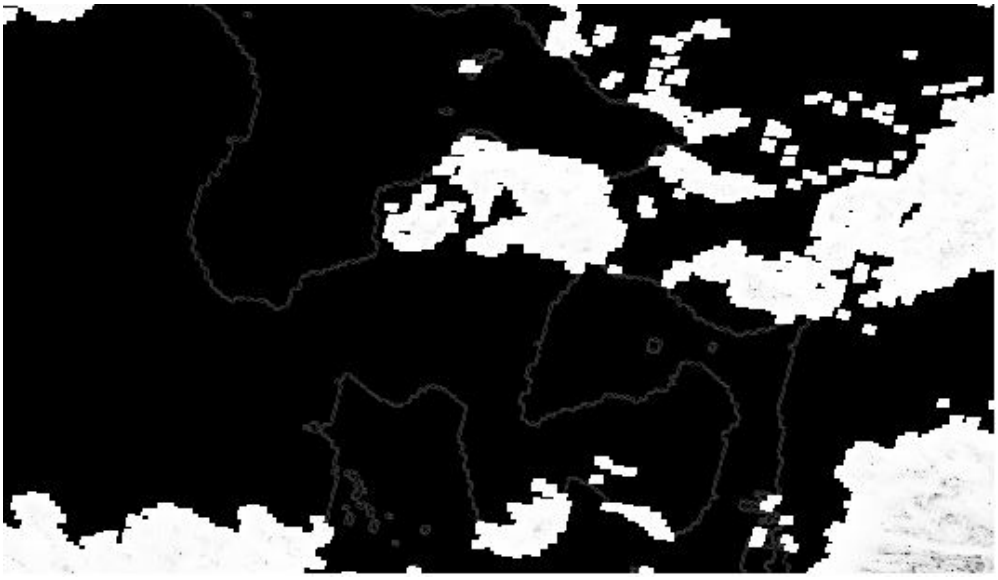

図 9.5　ラスタのベクタ化を行ったデータ

Vector.py について

Vector.py ではベクタ化したラスタデータのうち、一定の値以上のものを選択して保存する。

qgis:selectbyattribute による地物の選択

リスト 9.24　Vector.py(1)

```
processing.run('qgis:selectbyattribute',  {
                'INPUT':layer,#ベクタレイヤ
                'FIELD':  'DN',#選択するフィールド
                'OPERATOR':  0,
                'VALUE':  90,#検索する値
               'OUTPUT':  'G:/pdkx3.geojson'})#出力ファイル
```

qgis:selectbyattribute は入力レイヤから検索に一致した地物を選択する。検索するフィールドはデフォルトの DN から行い、最も傾斜が激しい部分が 90 となる地物を検索し選択する。

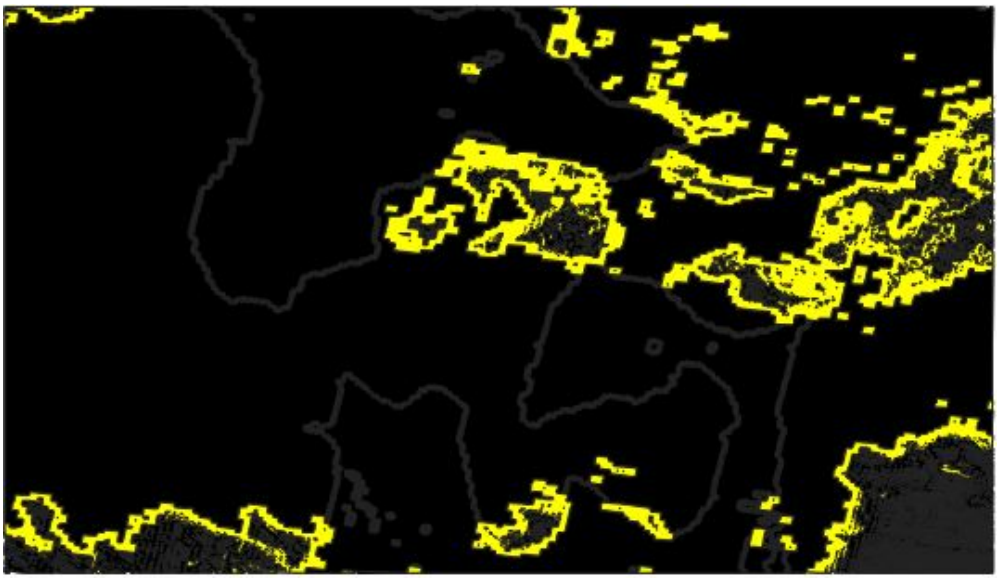

図 9.6　地物の選択をしたデータ

qgis:saveselectedfeatures による選択地物の保存

リスト 9.25　Vector.py(2)

```
processing.run('qgis:saveselectedfeatures',  {
                'INPUT':layer,#現在選択中のベクタレイヤ
               'OUTPUT':  'G:/pdkx5'+'.GeoJSON'})#出力ファイル
```

qgis:saveselectedfeatures は選択された地物から新しいレイヤを作成する。

（※文責: 蓬畑尚輝）

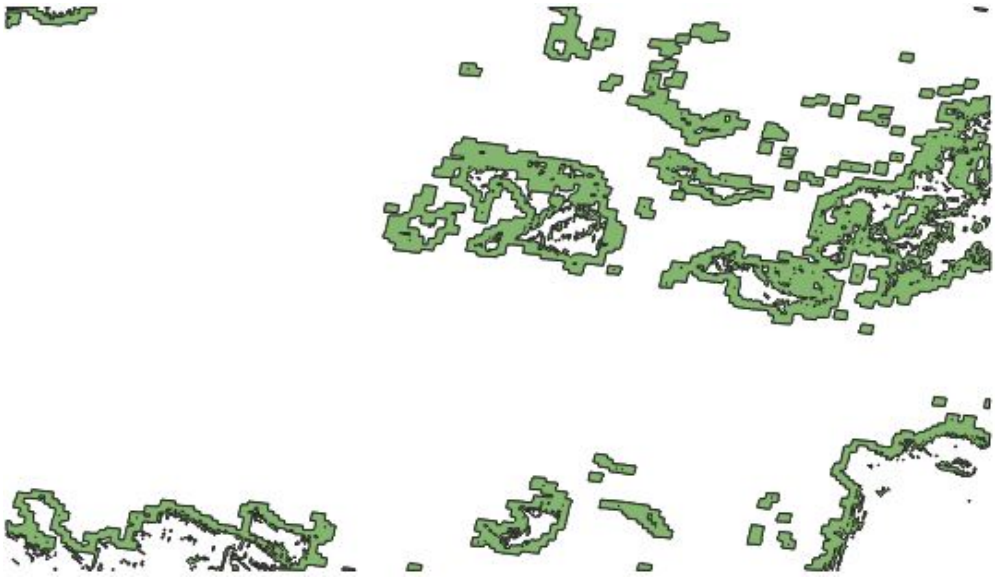

図 9.7　解析をした画像

第 10 章　今後の展望

10.1　フロントエンド

海グループの第 5 章の未解決の課題とアプローチと今回得た結果より、今後は UI を改善し、一目でどのような機能があるかをわかりやすくすること、検索機能や魚の釣り床の表示などの利用者が使いやすいような機能の追加、利用規約の作成をして利用者とのトラブルが起きた際に対応をしやすくすること、そしてもともと本グループの目標であるアプリをストアに対して審査をしてもらい配信することを展望とする。また、プログラムのリファクタリングも行いたい。リファクタリングとは、プログラム自体の外部的な挙動を変えずに内部の構造や挙動を改善することだ。現在、アプリケーションを実現するためのプログラムはクラスごとに役割を分け切れておらず、可読性が悪くなっている。この問題を解決するためにリファクタリングを行いたい。バックエンド側の解析ファイルを表示する方法の変更も行いたい。現在の表示方法では表示が重く、操作に支障がでる可能性がある。そのため、解析ファイルの表示方法を変更し、表示を軽くしたい。

（※文責: 山田純平）

10.2　バックエンド

10.2.1　バックエンドサーバの完全自動化

第 5 章のバックエンドの自動化で述べたとおり、サーバ上での自動化を目指した部分のうち、「サーバで QGIS を用いて自動で画像を解析する」という部分と、「G-Portal のサーバから目標となるデータのみを取得する」という部分の自動化が達成できなかった。これらの問題は技術のレベルが高く、また、参考にするための文献も少ないため、使用するツール自体を見直して別の角度から課題解決に向けてアプローチしていきたい。前者に関しての解決方法として、QGIS の Python 用 API である PyQGIS を使用せずに、地理データの加工用の Python ライブラリを使用して問題解決を目指したい。また、後者の問題に関しては解析元のデータを提供しているツールは G-Portal の他にいくつかあるため、それらのツールについて調べて、データを自動取得できそうか検討していきたい。

（※文責: 外川雄也）

10.2.2　リアルタイムのデータ取得

本プロジェクトでは解析する元となるデータの取得を行うサービスとして、サービスの扱いやすさとデータの取得のしやすさを考慮して、G-portal というサービスを利用している。しかしながら、G-portal で提供されているデータはデータを提供する前にすべて精度を検証する必要があり、そのために人工衛星によって観測された日から 1〜2 日程度も時間がかかってしまう。これによってアプリに表示されるデータは精度は高いが、観測から 1 日後のデータが表示され、リアルタイム

性が低くなってしまうという問題が生じている。観測からデータの提供までが遅いとその間に魚が取れやすい場所が変化してしまう可能性が生じてしまう。この問題を解決するために展望として、ある程度精度を保ちつつ、リアルタイム性が高いデータを提供する方法について考えていきたい。

（※文責: 外川雄也）

10.2.3　海面水温の他にクロロフィルα濃度の測定による探知精度の向上

衛生から提供されているデータには海水面温度の他にクロロフィルａ濃度がある。クロロフィルａ濃度とは、植物プランクトンが持っている光合成色素の濃度のことで、クロロフィルａ濃度はその場所の植物プランクトンの量を示している。植物プランクトンは他の動物プランクトンや魚のえさとなっているため、クロロフィルａの濃度から魚の豊富な漁場探索することができる。漁場の探索のために海水面温度の他にこのクロロフィルａ濃度を用いて海水面データを解析し、それらの解析結果を照合することによって、漁場の探索精度の向上を目指していきたい。

（※文責: 外川雄也）

付録 A　新規習得技術

Swift：アプリケーションのフロントエンドを作成するために使用した言語
Xcode：フロントエンドを作成するために使用した統合開発環境

（※文責: 山本陽平）

付録 B　活用した講義

・Python（情報処理演習 II）
・サーバ構築（データベース工学）

参考文献

[1] だいち（ALOS）―人工衛星プロジェクト｜ JAXA 第一宇宙技術部門 https://www.satnavi.jaxa.jp/ja/project/alos/ （参照 2022-01-19）

[2] 水産× IT http://marine-it.net/activity/ubiquitous-buoy.shtml（参照 2022-01-19）

[3] 産業別就業者数 https://www.jil.go.jp/kokunai/statistics/timeseries/html/g0204.html（参照 2022-01-19）

[4] 函館市の近況 https://www.city.hakodate.hokkaido.jp/docs/2020033100148/files/1.pdf（参照 2022-01-19）

[5] business leaders square wisdom https://wisdom.nec.com/ja/collaboration/2016120501/02.html（参照 2022-01-19）

[6] くらしごと https://kurashigoto.hokkaido.jp/life/20190819100000.php（参照 2022-01-19）

[7] 経済産業省、平成 23 年 3 月 10 日「リモートセンシングに係る現状と課題」(https://tinyurl.com/asbpc4d6)（参照 2022-01-19）

[8] 釣りドコ https://turidoco.com/areas/1995（参照 2022-01-19）

[9] QGIS プロジェクトへようこそ！.https://qgis.org/ja/site/（参照 2022-01-19）

[10] 国土交通省、国土地理院 https://maps.gsi.go.jp/development/siyou.html（参照 2022-01-19）

[11] World Health Organization https://www.who.int/health-topics/coronavirus#tab=tab_3（参照 2022-01-19）

[12] 公立はこだて未来大学 HP https://www.fun.ac.jp/news/7648（参照 2022-01-19）

[13] SE Design 「初心者でも分かる「Microsoft Teams」の基本〜超便利なビジネスコミュニケーションツールの機能と使い方」https://www.sedesign.co.jp/blog/how-to-use-teams（参照 2022-01-19）

[14] Coronavirus disease (COVID-19) https://www.who.int/health-topics/coronavirus#tab=tab_3（参照 2022-01-19）

[15] はこだて未来大学「投稿自粛について」 https://www.who.int/healthtopics/coronavirus#tab=tab_3（参照 2022-01-19）