

チーム開発

3138 松井三吉

目的

- 一つのシステムを複数人からなるチームで構築するポイントや流れを理解する
- 一つのシステム(システムは複数のプログラムの集合体)を複数人で実装する経験を積む
- git/githubのpull requestを使ったコードレビューとマージの概念を理解し、実践する
- 開発だけではなく、運用を想定し、開発環境や使うツールの選定や整備を経験する

使用するツール

- docker, docker-compose
- Node.js
- React
- socket.io
- MySQL
- SQLAlchemy
- marshmallow
- git
- github

課題 1 gitの練習




目的

数人で開発する時のソースコード等の共有をgithubで行う時の注意点やコツを理解する。ブランチは、ファイルの変更履歴の流れを分岐して記録していくためのものである。分岐したブランチは他のブランチに影響を与えない・影響を受けないため、複数の変更を同時並行して進めていくことができる。

手順

1. チームメンバーを確認する(3または4人)欠席がいた場合位は残り的人で頑張る(休んだ人は次週までに追いつく)。
2. チームリーダーを決める。リーダーはリンクからGitHub Campusでチームを作成し(チーム名はチーム番号)、チームのリポジトリを作成する。
3. チームリーダーはチームメンバーをリポジトリの開発者として追加する(他の人はアカウント名を教え、近くで見ていること)。
4. Branchに対するpush/pullの練習、pull requestの練習を行う

pull requestの結果

<input type="checkbox"/>		add	#3 by hyouga112 was merged on Dec 6, 2024
<input type="checkbox"/>		a	#2 by mooeka1 was merged on Dec 6, 2024
<input type="checkbox"/>		rugia	#1 by r04i30 was merged on Dec 6, 2024

更新場所が重複しない場合はコンフリクトが起こらず重複した場合は起こった

pull requestにより何が実現できているか

複数人が同時に作業することができ同じ箇所を変更した場合にもbranchで分けているのでそのタイミングでは円滑に開発がすすめられ最終的にMargeのタイミングでMargeする人がコンフリクトを解消することができる。また各部の開発部分の変更を見やすいため間違いに気づきやすくなる。

課題2 チーム開発

共同開発者

番号	名前
22	竹野萌花
27	野田飛翔
31	廣瀬充希

担当パート

Marge役（チームリーダー）

担当パートの確認方法

- 1. Pull Requestを送ってもらう
- 2. コードを確認し間違いがないかを確認
- 3. コンフリクトが起こっているかを確認
 - コンフリクトが起こっていれば解消する
- 4. Margeする

担当パートの結果

🔗 0 Open ✓ 7 Closed

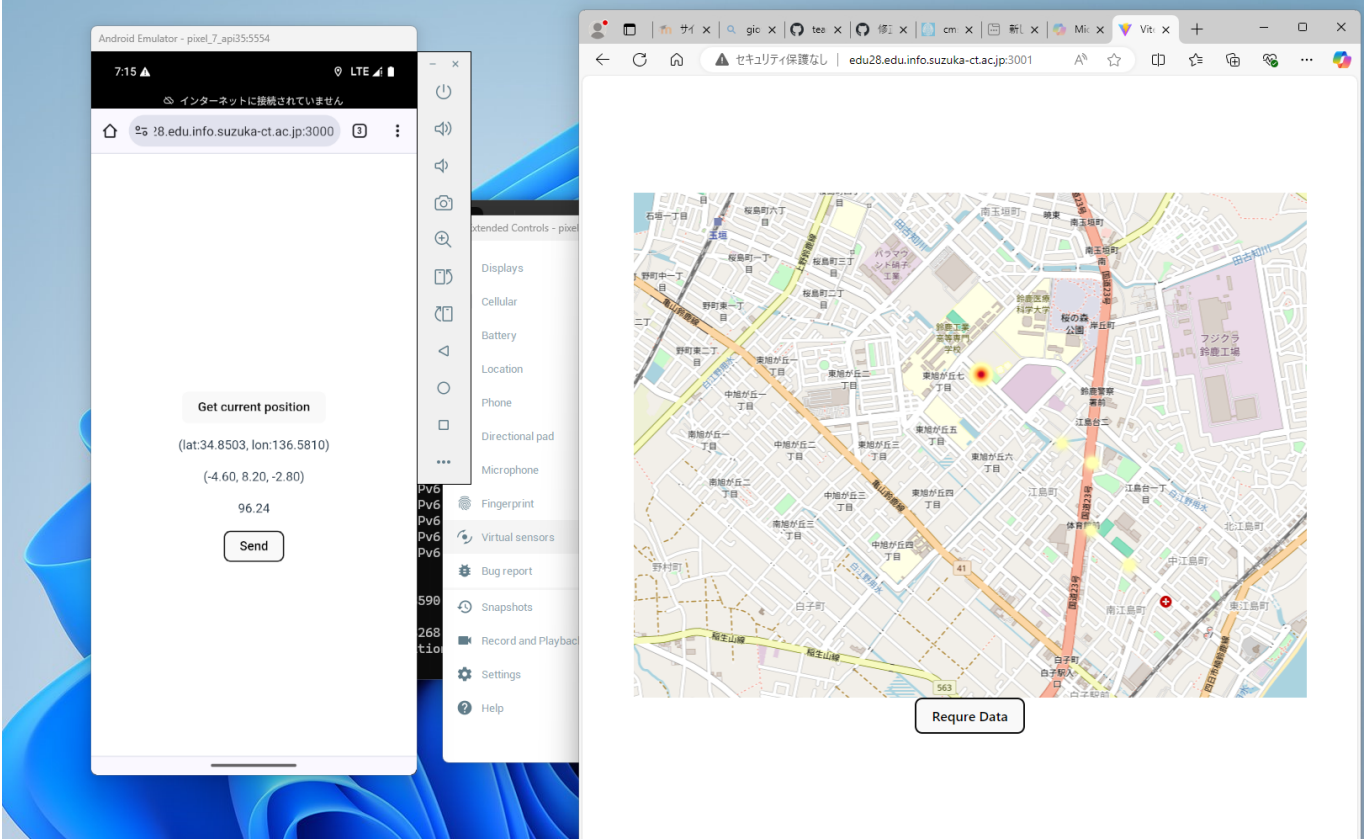
<input type="checkbox"/>	Author ▾	Label ▾	Projects ▾	Milestones ▾	Reviews ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	🔗 jj						
#7 by r04i30 was merged 3 weeks ago							
<input type="checkbox"/>	🔗 修正						
#6 by sansan2341 was merged 3 weeks ago							
<input type="checkbox"/>	🔗 server発展の手前まで完了						
#5 by mooea1 was merged last month							
<input type="checkbox"/>	🔗 communism						
#4 by r04i30 was merged last month							
<input type="checkbox"/>	🔴🔴 5までやってある(6から)						
#3 by hyouga112 was closed last month							
<input type="checkbox"/>	🔗 Sensor実装しました！！！！						
#2 by mooea1 was merged on Dec 6, 2024							
<input type="checkbox"/>	🔴🔴 sensor実装しました！！！！						
#1 by mooea1 was closed on Dec 6, 2024							

一度だけコンフリクトがcompose.ymlファイルで起きた
消すだけだったので消去し解決

チーム全体で作成したシステムの確認方法

- 1. それぞれのパートで確認作業を行う
- 2. すべてをMergeしたdev branchで動作するかを確認

チーム全体で作成したシステムの結果



エラーが起こったが修正し正しく動作した

エラー内容

送る情報が多すぎたエラーが起こった

エラー原因

修正前

```
@socketio.on('requiredata')
def requiredata():
    # DBから取得
    locdata = session.query(Locdata).all()
    # JSON <-> Object 変換スキーマ
    schema = LocdataSchema(many=True)
    # Object -> JSON -> JSON String に変換
    jsonobj = schema.dumps(locdata)
    jsonstring = json.dumps(jsonobj)
    # 'geoloc'メッセージで送る
    emit('geoloc', jsonstring)
    pass
```

修正後

```
@socketio.on('requiredata')
def requiredata():
```

```
# DBから取得
locdata = session.query(Locdata).all()
# JSON <-> Object 変換スキーマ
schema = LocdataSchema(many=True)
# Object -> JSON -> JSON String に変換
jsonobj = schema.dump(locdata)
jsonstring = json.dumps(jsonobj)
# 'geoloc'メッセージで送る
emit('geoloc',jsonstring)
pass
```

schema.dumpがschema.dumpsになっていたためにファイルに正しく書き込む処理が行われていなかった

実験を通じて気づいたチームで開発するときの注意点

ある程度開発したことがある人はスムーズに開発が進むがあまりしていない人はgitの使い方から始まるため初めての人と開発するときは気を付けなければならない。また、あまり考えずにその開発パートが終わっていないのにpull requestを送っている人がいたり pull requestを送る先を間違えていたりするのでMergeする際には細心の注意を払わなければならないと考える

理解したこと、理解できていないこと

gitの使い方を再確認できてあまりやったことのないMergeをできた。理解したこととしてよく見ないとエラーが分からないことと正確に確認しても見逃していることがあり後で苦しくなることを理解した。理解できなかったことは、VSCodeのエラーメッセージにエラーが出た部分にかかわることが出てこなかったことです。