



AI 개복치 게임

구현 영상: <https://www.youtube.com/watch?v=XeQBfq-We20>

Flutter 깃랩 : [https://gitlab.com/rkddbwns123/sunfish ai game](https://gitlab.com/rkddbwns123/sunfish_ai_game)

제작 기간 2023.03.02 – 2023.04.07
강유준

목차



1 기획 의도

2 소프트웨어 구성

3 Back-End

4 Front-End

5 쿠버네티스 배포

Part 1



기획 의도

기획 의도

요즘 같은 시대에 심심이나 이루다같은 AI 챗봇을 모르는 사람은 굉장히 드물 것이다.
특히나 ChatGpt라는 완성도 높은 AI 챗봇이 나온 현재, AI에 대한 관심은 더욱이 높아져만 가고 있다.
멀리서 봤을 땐 정말 신기하다, 어떻게 저렇게 완성도가 높을까 하고 감탄만 해됐지만, 개발을 배우면서 그 궁금증은 다른 쪽으로 흐르기 시작했다.
AI 챗봇이 탄생한 과정부터 머신 러닝, 딥 러닝 등의 교육 방식, 그리고 나도 저런 챗봇을 만들 수 있을까 하는 궁금증.
그래서 만들게 되었다.
구글의 ChatGpt나 이미 데이터가 많이 쌓인 챗봇들에 비교할 순 없겠지만, 중요한 건 이런 AI 챗봇들이 어떤 식으로 구현이 되었는가에 대한 지식이라고 생각한다.

Part 2



소프트웨어 구성



- Java
- SpringBoot
- JPA
- Postgres
- DialogFlow Api



- Dart
- Flutter

Part 3



Back-End


```

@ApiModelProperty(notes = "시퀀스")
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@ApiModelProperty(notes = "엔티티명")
@Column(nullable = false, length = 100)
private String dfeName;

@ApiModelProperty(notes = "엔티티 상태 / 긍정, 부정, 중립")
@Enumerated(value = EnumType.STRING)
@Column(nullable = false)
private ValueStatus valueStatus;

@ApiModelProperty(notes = "값 변화에 영향이 있는가")
@Column(nullable = false)
private Boolean isChangeValue;

```

DialogFlow Entity를 관리할 Entity 설정
Builder 패턴 사용

```

1 usage
public void putValueCount(ValueCountUpdateRequest request) {
    this.dfeName = request.getDfeName();
    this.valueStatus = request.getValueStatus();
    this.isChangeValue = request.getIsChangeValue();
}

```

```

public void putValueCount(ValueCountUpdateRequest request) {
    this.dfeName = request.getDfeName();
    this.valueStatus = request.getValueStatus();
    this.isChangeValue = request.getIsChangeValue();
}

1 usage
private ValueCount(ValueCountBuilder builder) {
    this.dfeName = builder.dfeName;
    this.valueStatus = builder.valueStatus;
    this.isChangeValue = builder.isChangeValue;
}

2 usages
public static class ValueCountBuilder implements CommonModelBuilder<ValueCount> {

    2 usages
    private final String dfeName;
    2 usages
    private final ValueStatus valueStatus;

    2 usages
    private final Boolean isChangeValue;

    1 usage
    public ValueCountBuilder(ValueCountRequest request) {
        this.dfeName = request.getDfeName();
        this.valueStatus = request.getValueStatus();
        this.isChangeValue = request.getIsChangeValue();
    }

    1 usage
    @Override
    public ValueCount build() { return new ValueCount( builder: this); }
}

```



```
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class DieCondition {
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ApiModelProperty(notes = "인텐트명")
    @Column(nullable = false, length = 100)
    private String intentName;
```

즉사 이벤트 발생 Intent 관리 Entity 설정 , Builder 패턴 사용

1 usage 강유준

```
public void putDieCondition(DieConditionUpdateRequest request) { this.intentName = request.getIntentName(); }
```

1 usage 강유준

```
private DieCondition(DieConditionBuilder builder) { this.intentName = builder.intentName; }
```

2 usages 강유준

```
public static class DieConditionBuilder implements CommonModelBuilder<DieCondition> {
```

2 usages

```
private final String intentName;
```

1 usage 강유준

```
public DieConditionBuilder(DieConditionRequest request) { this.intentName = request.getIntentName(); }
```

강유준

```
@Override
```

```
public DieCondition build() { return new DieCondition( builder: this); }
```

```
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class Member {
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ApiModelProperty(notes = "고객 명")
    @Column(nullable = false, length = 20)
    private String name;

    @ApiModelProperty(notes = "아이디")
    @Column(nullable = false, length = 20, unique = true)
    private String username;

    @ApiModelProperty(notes = "비밀번호")
    @Column(nullable = false, length = 100)
    private String password;
}
```

로그인 ID, 비밀번호 관리 Entity 설정

```
public int calculateChangeLifeCount(List<String> paramKeys) {  
    List<ValueCount> valueCounts = new LinkedList<>();  
  
    for (String key : paramKeys) {  
        Optional<ValueCount> tempResult = valueCountRepository.findByDfeName(key);  
        tempResult.ifPresent(valueCounts::add);  
    }  
  
    int weightValue = 0;  
    int weightCount = 0;  
    for (ValueCount valueCount : valueCounts) {  
        weightValue += valueCount.getValueStatus().getWeighted();  
        if (valueCount.getIsChangeValue()) {  
            weightCount += 1;  
        }  
    }  
    return weightValue * weightCount;  
}
```

DialogFlow Entity 별로 생명력을 깎는 계산식.

Entity 별로 점수를 책정해놓고 DialogFlow에서 받은 Entity 명과 같은 데이터를 Repository에서 받아오고 점수를 계산

Entity가 2개 이상일 경우 점수를 n 배로 계산하게 계산식 작성

```
1 usage  👤 강유준 *  
public GameResponse doCalculation(GameRequest request) {  
    DialogFlowResponse dialogFlowResponse = dialogFlowService.detectIntentTexts(request.getMessage());  
    long intentDuplicationCount = dieConditionRepository.countByIntentName(dialogFlowResponse.getIntentName());  
  
    if (intentDuplicationCount > 0) {  
        return new GameResponse.GameResponseBuilder(  
            dialogFlowResponse.getQueryText(), |  
            dialogFlowResponse.getFulfillmentText(),  
            isDead: true,  
            changeLifeCount: 0).build();  
    } else {  
        return new GameResponse.GameResponseBuilder(  
            dialogFlowResponse.getQueryText(),  
            dialogFlowResponse.getFulfillmentText(),  
            isDead: false,  
            calculateChangeLifeCount(dialogFlowResponse.getEntities()))).build();  
    }  
}
```

DialogFlow API에서 데이터를 받아온다.

받아온 Intent 이름과 같은 Intent를 Repository에서 찾고 만약 있다면 즉시 처리, 없다면 계산식에서 받아온 값만큼 생명력 차감.


```
public DialogFlowResponse detectIntentTexts(String callMsg) throws ApiException {  
    String projectId = "my-project-id";  
  
    String randomSessionId = UUID.randomUUID().toString();  
  
    String langCode = "ko-KR";  
  
    // Instantiates a client  
    try (SessionsClient sessionsClient = SessionsClient.create()) {  
        // Set the session name using the sessionId (UUID) and projectId (my-project-id)  
        SessionName session = SessionName.of(projectId, randomSessionId);  
  
        TextInput.Builder textInput = TextInput.newBuilder().setText(callMsg).setLanguageCode(langCode);  
  
        // Build the query with the TextInput  
        QueryInput queryInput = QueryInput.newBuilder().setText(textInput).build();  
  
        // Performs the detect intent request  
        DetectIntentResponse response = sessionsClient.detectIntent(session, queryInput);  
  
        QueryResult result = response.getQueryResult();  
    }  
}
```

DialogFlow Api 에서 데이터를 받아오는 서비스

```
@Getter
@Setter
public class DialogFlowResponse {
    private String queryText;

    private String fulfillmentText;

    private String intentName;

    private List<String> entities;
}
```

```
QueryResult result = response.getQueryResult();

DialogFlowResponse resultResponse = new DialogFlowResponse();

resultResponse.setQueryText(result.getQueryText());
resultResponse.setFulfillmentText(result.getFulfillmentText());
resultResponse.setIntentName(result.getIntent().getDisplayName());

List<String> dfEntities = new LinkedList<>();
Struct temp1 = result.getParameters();
Map<String, Value> temp2 = temp1.getFieldsMap();
temp2.forEach((key, value) -> {
    dfEntities.add(key);
});
resultResponse.setEntities(dfEntities);

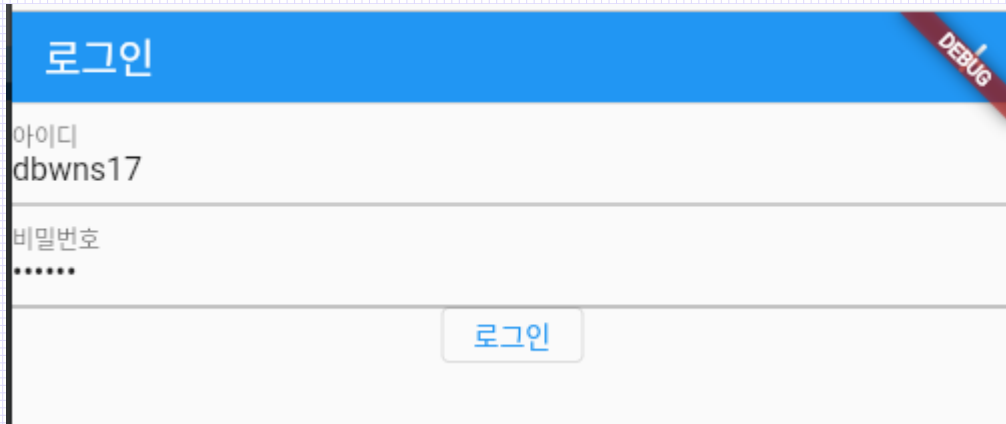
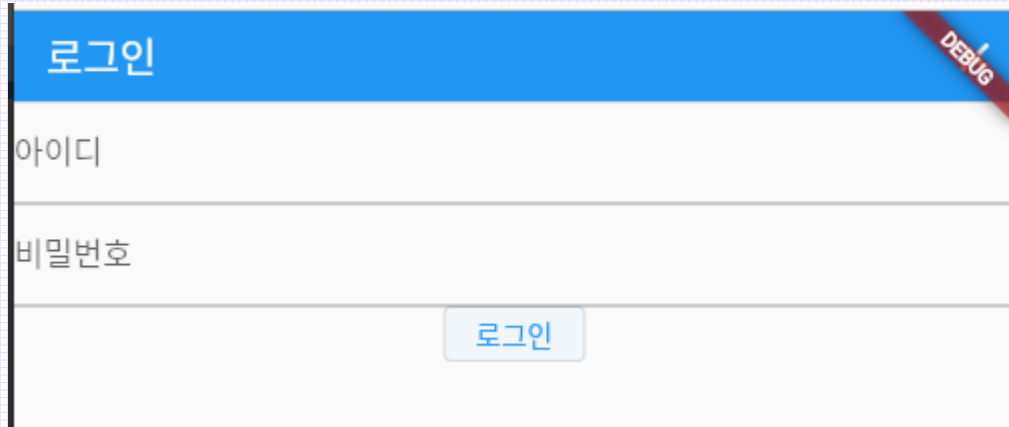
return resultResponse;
} catch (Exception e) {
    return null;
}
```

DialogFlowResponse 모델에 맞게 DialogFlowApi 에서 데이터를 받아준다.

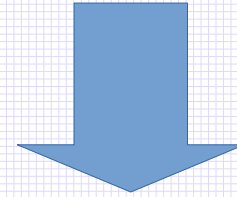
Part 4



Front-End

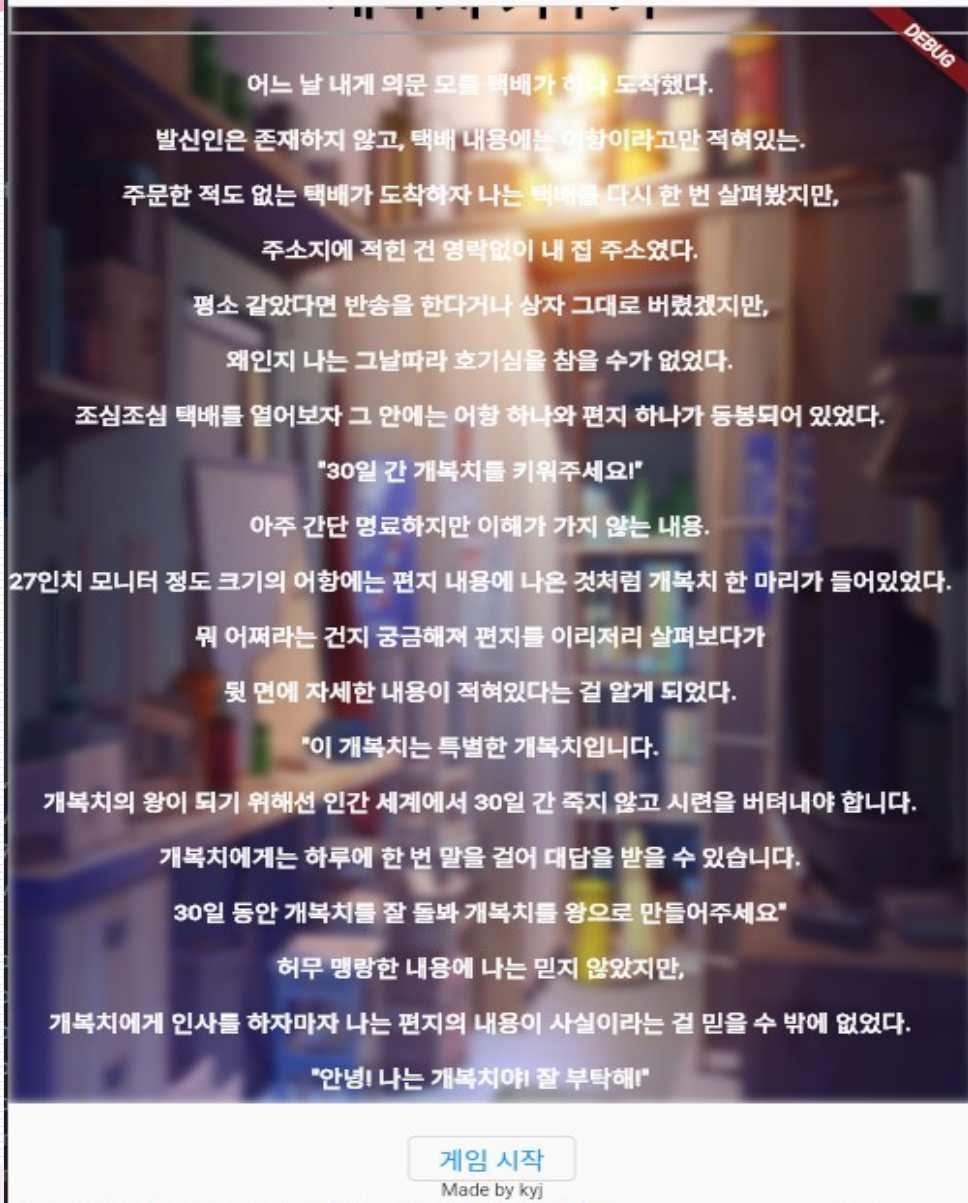


```
class _LoginCheckState extends State<LoginCheck> {  
  @override  
  void initState() {  
    super.initState();  
    MiddlewareLoginCheck().check(context);  
  }  
}
```



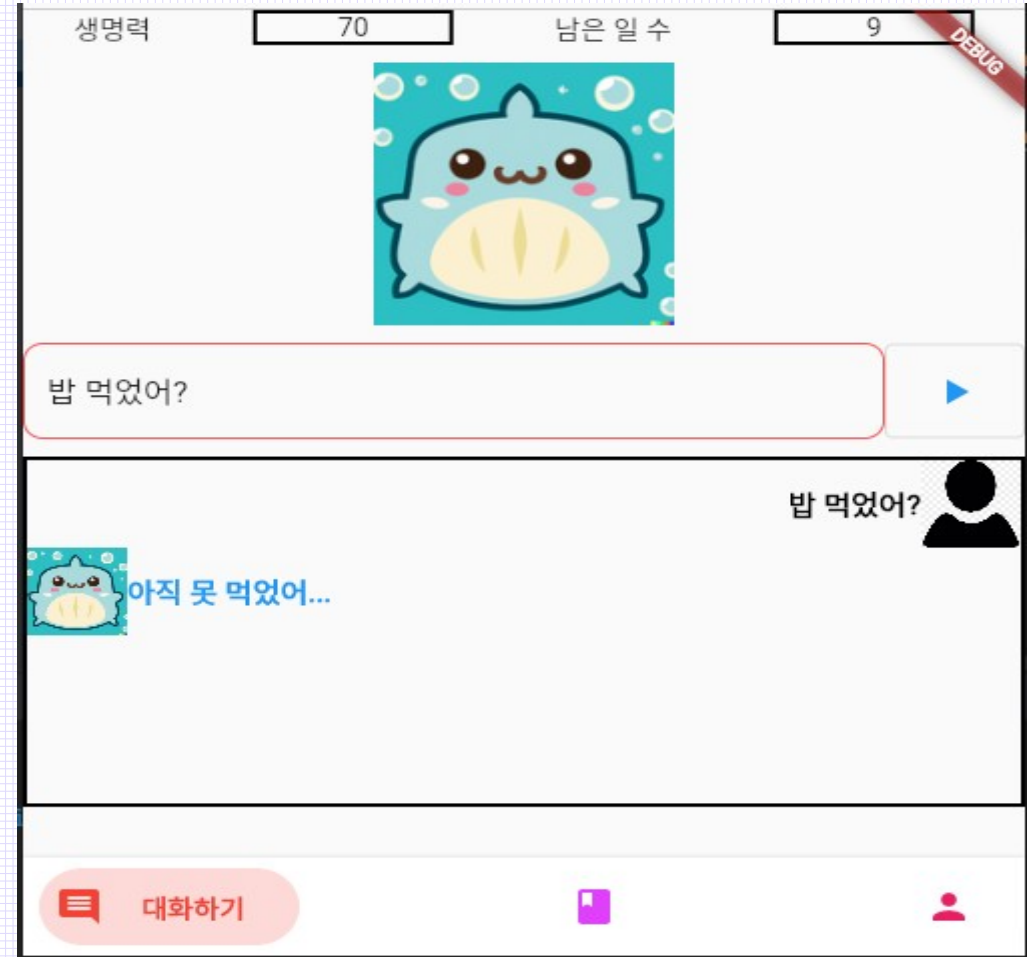
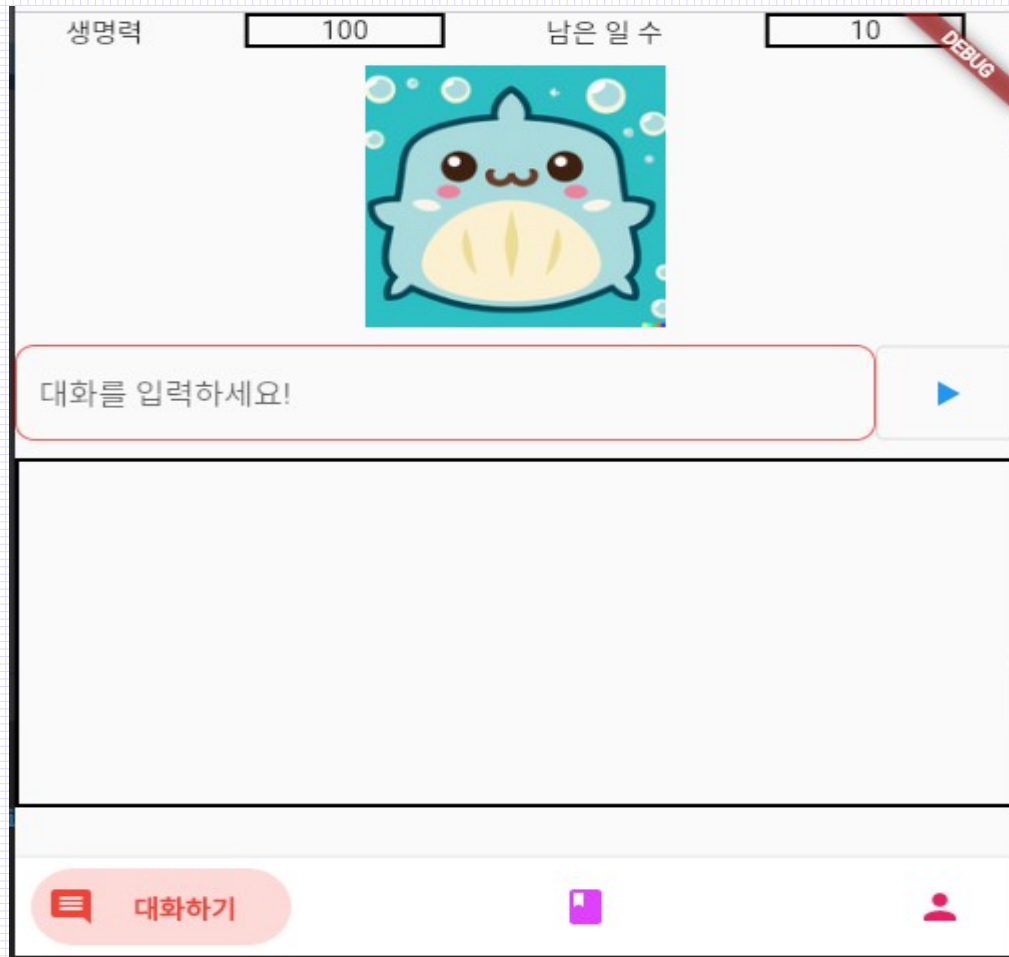
```
if (memberId == null) {  
  Navigator.pushAndRemoveUntil(context, MaterialPageRoute(  
    builder: (BuildContext context) => const PageLogin(), (route) => false);  
} else {  
  Navigator.pushAndRemoveUntil(context, MaterialPageRoute(  
    builder: (BuildContext context) => const PageStart(), (route) => false);  
}  
}
```

앱 시작 시 바로 로그인 상태를 확인하고 로그인이 되어 있다면 시작 페이지로 이동 / 되어 있지 않다면 로그인 화면 으로 이동



```
OutlinedButton(  
  onPressed: () {  
    Navigator.push(context, MaterialPageRoute(builder: (context) => const PageIndex()));  
  },  
  child: Text('게임 시작')  
) // OutlinedButton
```

MaterialPageRoute 를 이용해 게임 시작 버튼을 누르면
Index 페이지로 이동



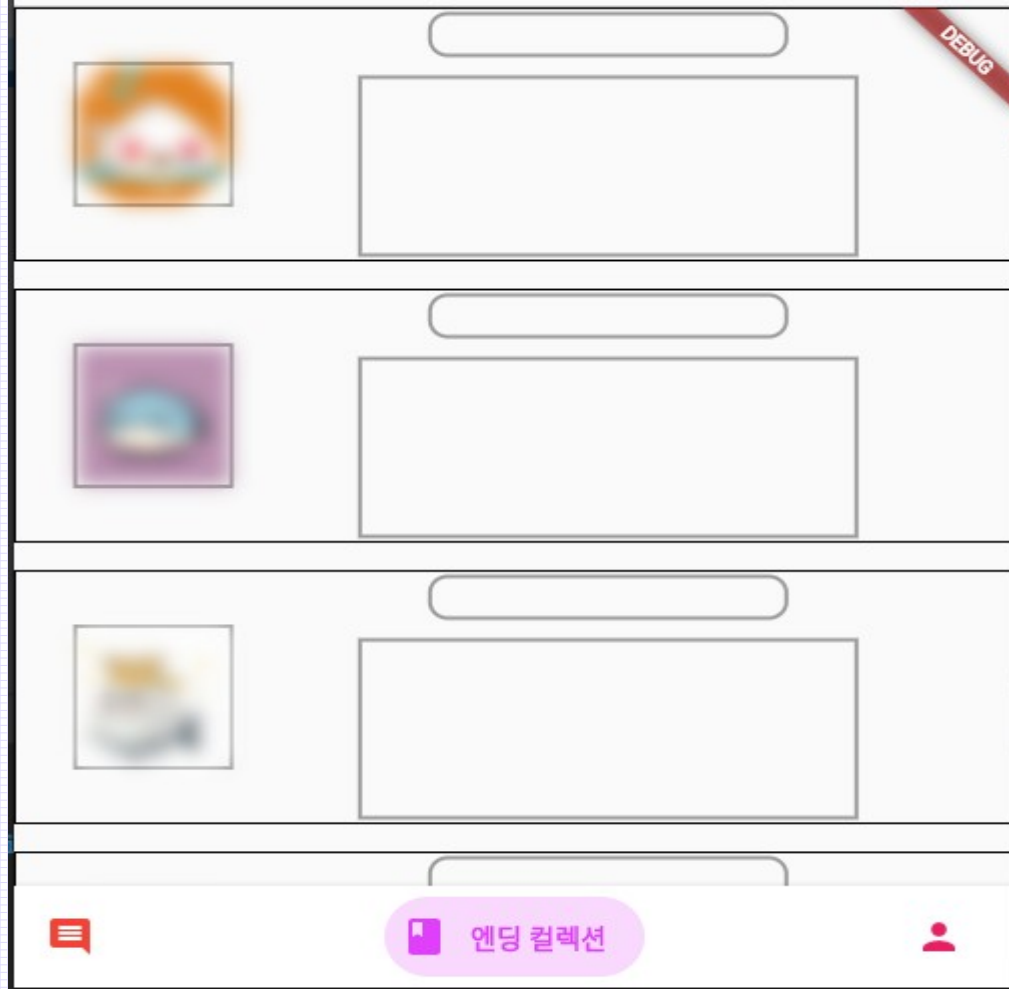
대화 입력 시 답변이 오고, 그 답변에 매겨진 생명력 점수를 차감(증가)시키고 남은 일 수를 줄인다.

```
Future<void> _sendMessage(String message) async {  
  BotToast.showCustomLoading(toastBuilder: (cancelFunc) {  
    return ComponentCustomLoading(cancelFunc: cancelFunc);  
  });  
  
  await RepoGame().sendMessage(message).then((res) {  
    BotToast.closeAllLoading();  
  
    setState(() {  
      _aiMessage = res.data.aiFulfillmentText;  
      _humanMessage = res.data.humanQueryText;  
      _lifeCount += res.data.changeLifeCount;  
      _remainDay -= 1;  
      _isDead = res.data.isDead;  
    });  
  
  }).catchError((err) {  
    ComponentNotification(  
      success: false,  
      title: '전송 실패',  
      subTitle: '전송에 실패하였습니다.',  
    ).call();  
  
    BotToast.closeAllLoading();  
  });  
}
```

```
String? _aiMessage;  
String? _humanMessage;  
int _lifeCount = 100;  
int _remainDay = 10;  
bool _isDead = false;  
bool _isGameEnd = false;
```

Api에서 데이터를 불러오는 중에 로딩창을 띄우고 불러오는 데 성공(실패) 시 닫는다.

실패 시 알림 창을 띄운다.



엔딩 컬렉션 / IF 문을 사용해 아직 보지 못한 엔딩에는 이미지를 블러 처리하고 정보를 숨겨놓았다.


```
String _getImageAddress(int endingNumber) {
    String result = '';
    if(endingNumber == 1) {
        result = 'assets/ending1.png';
    } else if(endingNumber == 2) {
        result = 'assets/ending2.png';
    } else if(endingNumber == 3) {
        result = 'assets/ending3.png';
    } else {
        result = 'assets/ending4.png';
    }
    return result;
}

String _getEndingText(int endingNumber) {
    String resultContent = '';
    if(endingNumber == 1) {
        resultContent = '개복치가 과도한 스트레스로 인해 사망하였습니다.';
    } else if(endingNumber == 2) {
        resultContent = '개복치가 시련을 견디지 못하고 끝 없는 잠에 빠졌습니다.';
    } else if(endingNumber == 3) {
        resultContent = '개복치가 개복치 왕이 되어 떠났습니다!';
    } else {
        resultContent = '개복치가 30일 간 버텨내긴 했지만, 개복치 왕이 되진 못했습니다.';
    }
}
```

```
body: ListView(
  children: [
    ComponentCollection(imageAddress: 'assets/ending1.png', endingTit
    SizedBox(height: 15,),
    ComponentCollection(imageAddress: 'assets/ending2.png', endingTit
    SizedBox(height: 15,),
    ComponentCollection(imageAddress: 'assets/ending3.png', endingTit
    SizedBox(height: 15,),
    ComponentCollection(imageAddress: 'assets/ending4.png', endingTit
  ],
) // ListView
```

엔딩 번호에 따라 이미지 파일과 엔딩 내용을 가져온다.

엔딩 컬렉션은 컴포넌트로 처리

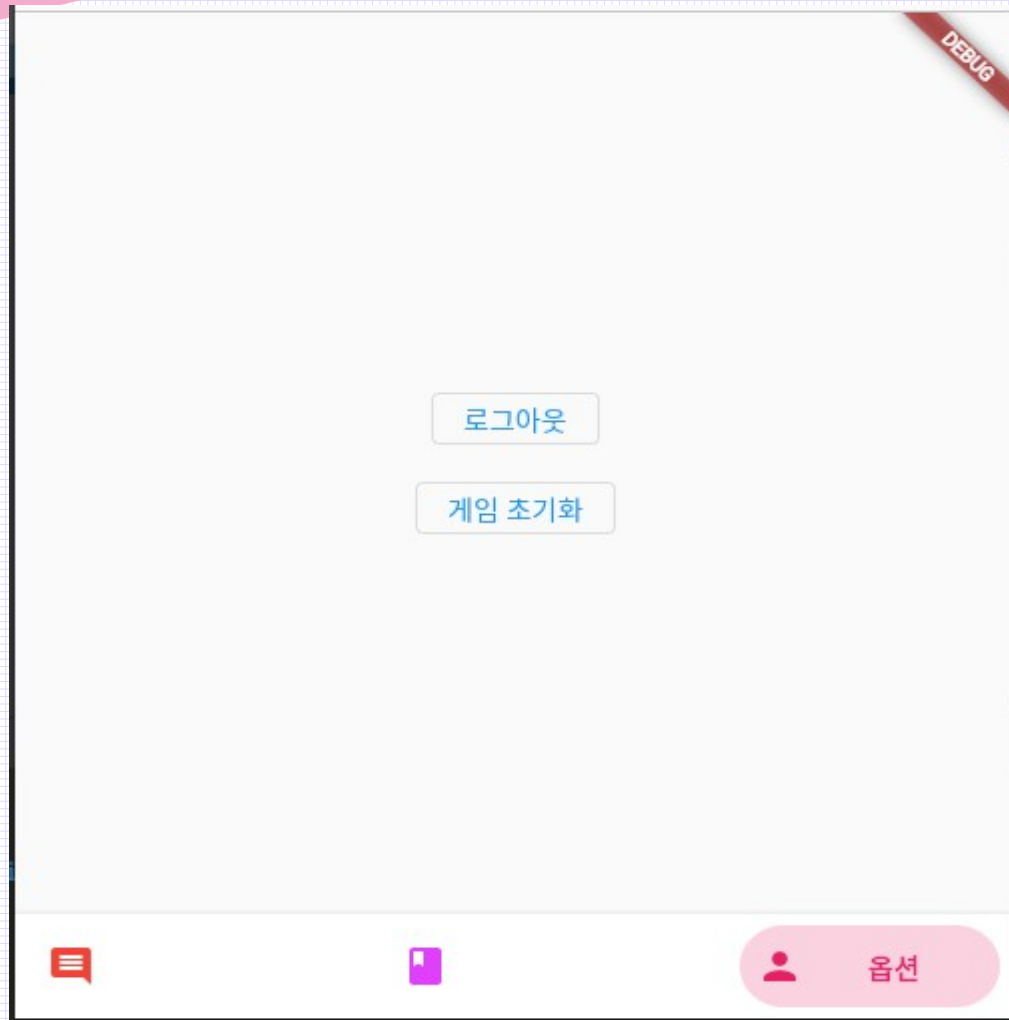
```
class _PageCollectionState extends State<PageCollection> {  
  bool _isEnding1 = false;  
  bool _isEnding2 = false;  
  bool _isEnding3 = false;  
  bool _isEnding4 = false;  
  
  void _initEndings() async {  
    final prefs = await SharedPreferences.getInstance();  
    bool? e1 = prefs.getBool('ending1');  
    bool? e2 = prefs.getBool('ending2');  
    bool? e3 = prefs.getBool('ending3');  
    bool? e4 = prefs.getBool('ending4');  
  
    setState(() {  
      _isEnding1 = e1!;  
      _isEnding2 = e2!;  
      _isEnding3 = e3!;  
      _isEnding4 = e4!;  
    });  
  }  
}
```

컬렉션에 처음 들어올 때(initState / 라이프사이클) SharedPreferences를 이용해 메모리에 저장된 데이터를 불러온다.



```
Future<void> _dialogEnding(int endingNumber) async {  
  String endingText = _getEndingText(endingNumber);  
  return showDialog<void>(  
    context: context,  
    barrierDismissible: false,  
    builder: (BuildContext context) {  
      return AlertDialog(  
        title: Text('엔딩 `${endingNumber}`'),  
        content: Column(  
          mainAxisAlignment: MainAxisAlignment.start,  
          children: [  
            SizedBox(  
              width: 100,  
              height: 100,  
              child: Image.asset(_getImageAddress(endingNumber)),  
            ), // SizedBox  
            if(endingNumber <= 2) Text('육성 실패') else Text('육성 성공'),  
            Text(endingText)  
          ],  
        ), // Column  
      ),  
    ),  
  );  
}
```

엔딩 조건 충족 시 엔딩 다이얼로그를 띄운다.



로그아웃 기능과 게임 초기화 기능



```
void _resetData() async{  
  final prefs = await SharedPreferences.getInstance();  
  prefs.clear();  
  prefs.setBool('ending1', false);  
  prefs.setBool('ending2', false);  
  prefs.setBool('ending3', false);  
  prefs.setBool('ending4', false);  
}  
  
Future<void> _logout(BuildContext context) async {  
  TokenLib.logout(context);  
}
```

메모리에 저장된 데이터를 불러오고 데이터를 초기화 시키고, 엔딩 컬렉션을 다시 false로 처리


```
class RepoMember {  
  Future<LoginResult> doLogin(LoginRequest loginRequest) async {  
    const String baseUrl = '$apiUri/member/login';  
  
    Dio dio = Dio();  
  
    final response = await dio.post(  
      baseUrl,  
      data: loginRequest.toJson(),  
      options: Options(  
        followRedirects: false,  
        validateStatus: (status) {  
          return status == 200;  
        }  
      )  
    );  
  
    return LoginResult.fromJson(response.data);  
  }  
}
```

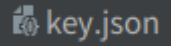
```
class RepoGame {  
  Future<SendMessageResponse> sendMessage(String message) async {  
    SendMessageRequest request = SendMessageRequest(message);  
    const String baseUrl = '$apiUri/game/send';  
  
    Dio dio = Dio();  
  
    final response = await dio.post(  
      baseUrl,  
      data: request.toJson(),  
      options: Options(  
        followRedirects: false,  
        validateStatus: (status) {  
          return status == 200;  
        }  
      )  
    );  
  
    return SendMessageResponse.fromJson(response.data);  
  }  
}
```

Dio를 사용해 외부 Api 를 사용, 데이터를 가져온다 (Json)

Part 5



쿠버네티스 배포

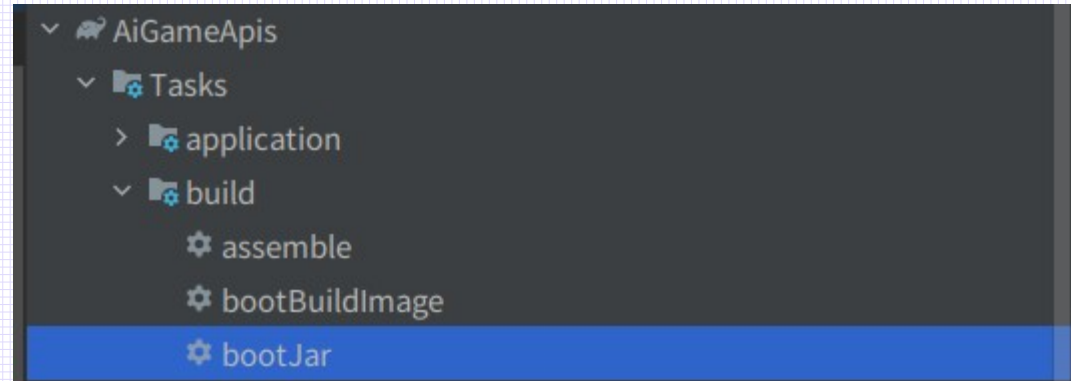


구글 서비스 계정 key를 json형태로 받아 넣는다.

```
FROM openjdk:17
RUN mkdir /build
ADD build/libs/*.jar /build/app.jar
EXPOSE 8080

ENTRYPOINT ["java", "-jar", "/build/app.jar"]
```

모듈 안에 Docker 파일 작성 (Dockerfile)



Jar 파일 생성

```
#!/bin/bash

gcpArea="asia.gcr.io"
projectId="██████████"
gcpRegion="us-central1"
gkeClusterName="autopilot-test"
dockerImage="api-game-logic"

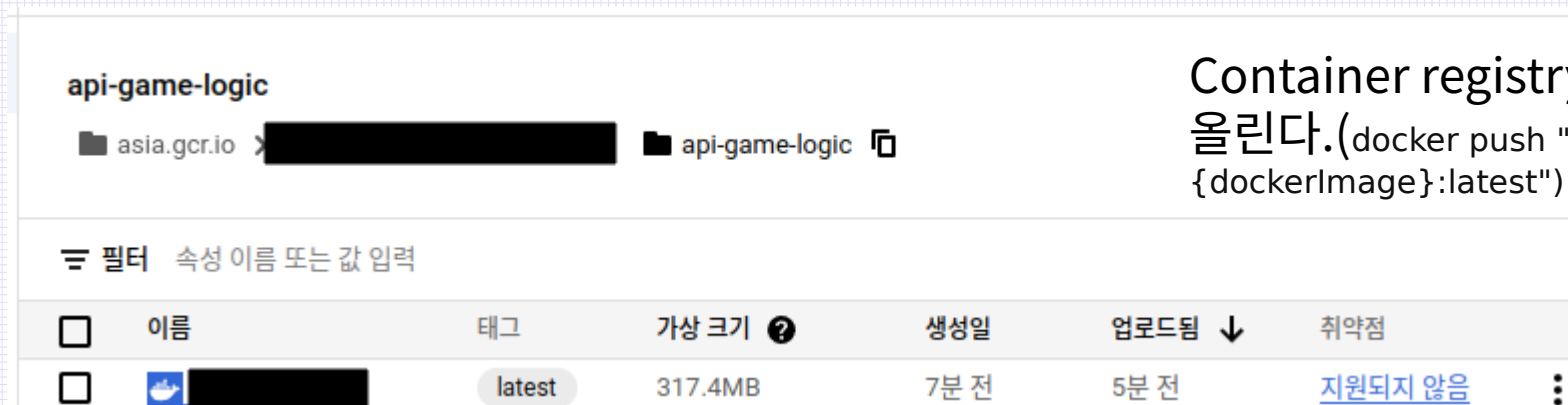
docker login -u _json_key --password-stdin https://asia.gcr.io < key.json
docker build -t ${dockerImage} .
docker tag ${dockerImage} "${gcpArea}/${projectId}/${dockerImage}"
docker push "${gcpArea}/${projectId}/${dockerImage}:latest"

export USE_GKE_GCLOUD_AUTH_PLUGIN=True
gcloud container clusters get-credentials ${gkeClusterName} --region ${gcpRegion} --project ${projectId}
kubectl delete deployment ${dockerImage}
kubectl apply -f k8s-deployment.yaml
kubectl apply -f k8s-service.yaml
kubectl apply -f k8s-ingress.yaml
```

uploadRelease.sh shell파일 작성 // 퍼미션 755로 바꿔준다.



내부 IP 생성



Container registry에 도커 image 파일을 올린다. (docker push "\${gcpArea}/\${projectId}/\${dockerImage}:latest")

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-game-logic
  labels:
    app: api-game-logic
spec:
  replicas: 1
  selector:
    matchLabels:
      app: api-game-logic
  template:
    metadata:
      labels:
        app: api-game-logic
    spec:
      containers:
        - name: api-game-logic
          image: asia.gcr.io/[redacted]/api-game-logic
          imagePullPolicy: Always
          resources:
            limits:
              cpu: "0.5"
              memory: "500Mi"
            requests:
              cpu: "0.5"
              memory: "500Mi"
          ports:
            - containerPort: 8080
          env:
            - name: spring.datasource.url
              value: jdbc:postgresql://[redacted]/ai-game-kyj
            - name: spring.datasource.username
              value: [redacted]
            - name: spring.datasource.password
              value: [redacted]
```

K8s.deployment.yaml 파일을
만든 후 쿠버네티스 deployment
생성.

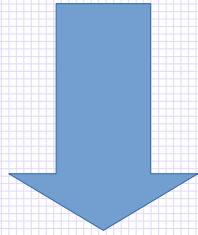
Env 환경 변수를 내부 ip 주소, 구글
클라우드 데이터 베이스로 변환
(배포용)

kubectl apply -f k8s-deployment.yaml
실행

≡ 필터 시스템 객체 : False ✕ 작업 부하 필터링 ✕ ? ≡

<input type="checkbox"/>	이름 ↑	상태	유형	pod	네임스페이스
<input type="checkbox"/>	api-game-logic	! Does not have minimum availability	Deployment	0/1	default

Deployment 확인
Auto-pilot 클러스터가 확인하고 pod 생성



≡ 필터 시스템 객체 : False ✕ 작업 부하 필터링 ✕ ? ≡

<input type="checkbox"/>	이름 ↑	상태	유형	pod	네임스페이스	클러스터
<input type="checkbox"/>	api-game-logic	✓ OK	Deployment	1/1	default	autopilot-test


```
apiVersion: v1
kind: Service
metadata:
  name: api-game-logic-service
spec:
  selector:
    app: api-game-logic
  ports:
    - port: 80
      targetPort: 8080
  type: ClusterIP
```

K8s.service.yaml 파일
Port =// http = 80
Https = 443
Http기 때문에 80으로 설정

<input type="checkbox"/>	이름 ↑	상태	유형	엔드포인트	pod	네임스페이스	
<input type="checkbox"/>	api-game-logic-service	✓ OK	클러스터 IP	10.9.128.252	1/1	default	▼

kubectl apply -f k8s-service.yaml 터미널 입력
구글 클라우드 쿠버네티스 엔진(서비스)에서 확인.


```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: api-game-logic-ingress
  annotations:
    kubernetes.io/ingress.class: "gce"
spec:
  rules:
  - host: "ai.usehwa-gen.shop"
    http:
      paths:
      - pathType: ImplementationSpecific
        path: "/"
        backend:
          service:
            name: api-game-logic-service
            port:
              number: 80
```

K8s.ingress.yaml 파일

host에 나의 도메인 주소 입력
전에 설정한 서비스와 포트 번호 입력

kubectl apply -f k8s-ingress.yaml

쿠버네티스 엔진(인그레스)에서 확인

<input type="checkbox"/>	이름 ↑	상태	유형	프런트엔드	서비스	네임스페이스	클러스터
<input type="checkbox"/>	api-game-logic-ingress	 Cre...	외부 HTTP(S) 부하 분산기	ai.usehwa-gen.shop/ ↗	api-ga... ▼	default	autopil... ▼



호스트 IP(A레코드) 추가

도메인/호스트 | ai.usehwa-gen.shop

IP주소 |

확인 >

마지막으로 도메인에서 호스트를 추가해준다.
IP 주소에는 ingress에서 할당받은 ip 주소를 입력

```
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$ gcloud container clusters get-credentials autopilot-test --region us-central1 --project [redacted]
Fetching cluster endpoint and auth data.
kubeconfig entry generated for autopilot-test.
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$ kubectl apply -f k8s-deployment.yaml
E0407 12:30:46.160653    8646 memcache.go:287] couldn't get resource list for metrics.k8s.io/v1beta1: the server is currently unable to handle the request
E0407 12:30:46.349761    8646 memcache.go:121] couldn't get resource list for metrics.k8s.io/v1beta1: the server is currently unable to handle the request
Warning: Autopilot increased resource requests for Deployment default/api-game-logic to meet requirements. See http://g.co/gke/autopilot-resources
deployment.apps/api-game-logic created
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$
```

```
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$ kubectl apply -f k8s-service.yaml
service/api-game-logic-service created
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$
```

```
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$ kubectl apply -f k8s-ingress.yaml
ingress.networking.k8s.io/api-game-logic-ingress created
dev-kyj@devkyj-desktop:~/workspace/java/AiGameApis/api-game-logic$
```

차례대로 deployment → service → ingress

ChatGPT



Examples

"Explain quantum computing in simple terms" →

"Get any creative ideas for a 10 year old's birthday?" →

"How do I make an HTTP request using Python?" →



Capabilities

Remembers what user said earlier in the conversation

Allows user to provide follow-up corrections

Trained to decline inappropriate requests



Limitations

May occasionally generate incorrect information

May occasionally produce harmful instructions or biased content

Limited knowledge: it only knows information up to 2021

감사합니다