v.0.0.2

최병권

광석 캐기 게임

목차 table of contents

Part 1

기획의도

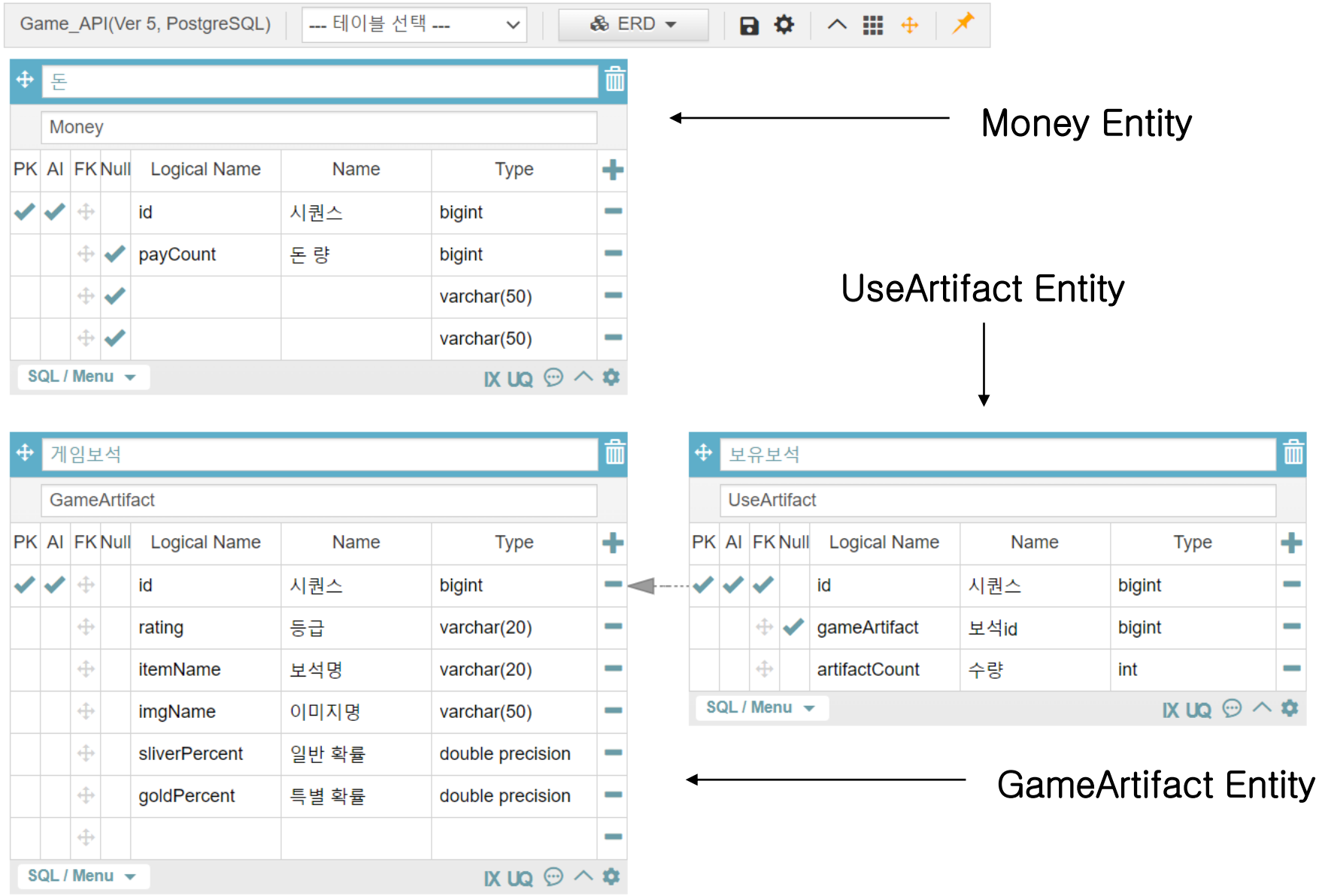
기획 의도

요즘 현대인들이 많이 즐기는 이른바 ‘클릭 게임’을 구현 해보기 위해 만들어 보기로 결정했다.
별 의미없이 클릭만 하고 돈을 모아서 컬렉션을 수집하는 것이지만 온라인 게임에서는 그것을 모으려고 많은 시간을 투자한다.
단순히 공을 들여 컬렉션 목록에 있는 아이템 들을 모두 수집하는 것이지만 그것을 모두 모으면 조그마한 성취감을 느낄 수 있다.
그러한 재미를 느끼기 위해 적당한 시간을 투자해야만 컬렉션 수집을 완료할 수 있는 게임을 만들었다.

Part 2

ERD 설계

AQUERY TOOL



- java 17
- Spring Boot
- Postgres

- Vue.js
- javascript
- Nuxt

Part 3

Front-End

template (1)

```
<template>
    <div class="full_frame">
        <el-row :gutter="20" class="mb-4" v-if="payCount != null">
            <el-col :span="8" class="contents-center">
                <div>
                    <h1 class="custom-title2">광산</h1>
                    <h2 class="custom-title1">현재 보유한 돈 : {{ payCount | currency }}원</h2>
                </div>
                <div class="mine">
                    <div class="dig_stone"><img src="/images/coal_mining.png" alt="광석캐기" width="500" height="308.5"/>
                    </div>
                    <div class="dig_button">
                        <el-button type="primary" round @click.native="getMoney()">광석 캐기! (10원)</el-button>
                    </div>
                </div>
            </el-col>
            <el-col :span="16">
                <div class="contents-center1">
                    <h1 class="custom-title2">뽑기</h1>
                </div>
                <div>
                    <el-row :gutter="20">
                        <el-col :span="12" class="contents-center">
                            <div class="sliver_box"><img src="/images/sliver.jpg" alt="은상자" width="300" height="300"/>
                            </div>
                            <div class="normal_dig">
                                <el-button type="warning" round @click.native="chooseBoxSilver()">일반 채굴 (100원)
                                </el-button>
                            </div>
                        </el-col>
                        <el-col :span="12" class="contents-center">
                            <div class="gold_box"><img src="/images/gold.jpg" alt="금상자" width="300" height="300"/></div>
                            <div class="special_dig">
                                <el-button type="danger" round @click.native="chooseBoxGold()">특별 채굴 (1,000원)
                                </el-button>
                            </div>
                        </el-col>
                    </el-row>
                </div>
```

```
            </el-col>
        </el-row>
        <div class="artifact-list-box" v-if="artifactList.length > 0">
            <h1 class="jewellery">보석 컬렉션</h1>
            <el-row :gutter="20">
                <el-col :span="4" class="contents-center" v-for="(item, index) in artifactList" v-bind:key="index">
                    <div v-if="item.artifactCount > 0">
                        <div><img :src="'/images/' + item.imageName" :alt="item.itemName" width="100" height="100"/>
                        </div>
                        <div>
                            <el-badge :value="item.ratingName" class="item" :type="getArtifactBadgeType(item.rating)">
                                {{ item.itemName }}
                            </el-badge>
                        </div>
                        <div>{{ item.artifactCount }}개</div>
                    </div>
                    <div v-else>
                        <div><img src="/images/artifact_no_img.jpg" alt="보석정보없음" width="100" height="100"/></div>
                        <div>
                            <el-badge :value="모름" class="item">
                                보석정보없음
                            </el-badge>
                        </div>
                        <div>-개</div>
                    </div>
                </el-col>
            </el-row>
        </div>

        <div class="reset_button">
            <el-button type="danger" round @click.native="gameReset()">게임리셋</el-button>
        </div>
```

template (2)

```
        <div class="reset_button">
            <el-button type="danger" round @click.native="gameReset()">게임리셋</el-button>
        </div>

        <el-dialog title="보석 뽑기 결과" :visible.sync="artifactDialogVisible" width="30%" center>
            <div v-if="currentChooseArtifact != null">
                <div class="contents-box1">다음 보석을 획득하셨습니다!</div>
                <div class="contents-center">
                    <div>
                        <img :src="'/images/' + currentChooseArtifact.imageName" :alt="currentChooseArtifact.itemName"
                             width="100" height="100"/>
                    </div>
                    <div>{{ currentChooseArtifact.ratingName }}</div>
                    <div>{{ currentChooseArtifact.itemName }}</div>
                    <div class="mt-4" v-if="currentChooseArtifact.isNew"><span class="custom-badge">NEW!!</span></div>
                </div>
            </div>

            <span slot="footer" class="dialog-footer">
                <el-button type="primary" @click="artifactDialogVisible = false">확인</el-button>
            </span>
        </el-dialog>
    </div>
</template>
```

Script (1)

```
<script>
no usages  Choi Byeonggwon *
export default {
    data() {
        return {
            payCount: null,
            artifactList: [],
            artifactDialogVisible: false,
            currentChooseArtifact: null
        }
    },
    methods: {
        getArtifactBadgeType(ratingValue) {
            let result = ''

            switch (ratingValue) {
                case 'NORMAL' :
                    result = 'info'
                    break
                case 'RARE' :
                    result = 'warning'
                    break
                case 'UNIQUE' :
                    result = 'primary'
                    break
                case 'LEGENDARY' :
                    result = 'danger'
                    break
                default:
                    result = ''
            }
            return result
        },
```

```
getFirstData() {
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
    this.$store.dispatch(this.$gameApiConstants.DO_FIRST_DATA)
        .then((res) => {
            this.payCount = res.data.data.moneyResponse.payCount
            this.artifactList = res.data.data.useArtifactItems
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
        .catch((err) => {
            this.$toast.error(err.response.data.msg)
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
},
```

Script (2)

```
getMoney() {
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
    this.$store.dispatch(this.$gameApiConstants.DO_SHOW_ME_THE_MONEY)
        .then((res) => {
            this.payCount = res.data.data.payCount
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
        .catch((err) => {
            this.$toast.error(err.response.data.msg)
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
},
chooseBoxSilver() {
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
    this.$store.dispatch(this.$gameApiConstants.DO_CHOOSE_BOX_SILVER)
        .then((res) => {
            this.payCount = res.data.data.moneyResponse.payCount
            this.artifactList = res.data.data.useArtifactItems
            this.currentChooseArtifact = res.data.data.currentItem
            this.artifactDialogVisible = true
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
        .catch((err) => {
            this.$toast.error(err.response.data.msg)
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
},
```

```
        chooseBoxGold() {
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
            this.$store.dispatch(this.$gameApiConstants.DO_CHOOSE_BOX_GOLD)
                .then((res) => {
                    this.payCount = res.data.data.moneyResponse.payCount
                    this.artifactList = res.data.data.useArtifactItems
                    this.currentChooseArtifact = res.data.data.currentItem
                    this.artifactDialogVisible = true
                    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
                })
                .catch((err) => {
                    this.$toast.error(err.response.data.msg)
                    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
                })
        },
        gameReset() {
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
            this.$store.dispatch(this.$gameApiConstants.DO_GAME_RESET)
                .then((res) => {
                    this.getFirstData()
                })
                .catch((err) => {
                    this.$toast.error(err.response.data.msg)
                    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
                })
        }
    },
    created() {
        this.getFirstData()
    },
    mounted() {
        this.$store.commit(this.$menuConstants.FETCH_SELECTED_MENU, payload: 'DASH_BOARD')
    },
}
</script>
```

Part 4

Back-End

GameArtifact Entity (1)

```java
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class GameArtifact {
    no usages
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    1 usage
    @ApiModelProperty(notes = "등급")
    @Column(nullable = false, length = 10)
    @Enumerated(value = EnumType.STRING)
    private Rating rating;

    1 usage
    @ApiModelProperty(notes = "유물명")
    @Column(nullable = false, length = 20)
    private String itemName;

    1 usage
    @ApiModelProperty(notes = "이미지명")
    @Column(nullable = false, length = 50)
    private String imageName;

    1 usage
    @ApiModelProperty(notes = "은상자 확률")
    @Column(nullable = false)
    private Double silverPercent;

    1 usage
    @ApiModelProperty(notes = "금상자 확률")
    @Column(nullable = false)
    private Double goldPercent;

    1 usage
    private GameArtifact(GameArtifactBuilder builder) {
        this.rating = builder.rating;
        this.itemName = builder.itemName;
        this.imageName = builder.imageName;
        this.silverPercent = builder.silverPercent;
        this.goldPercent = builder.goldPercent;
    }

    2 usages
    public static class GameArtifactBuilder implements CommonModelBuilder<GameArtifac
        2 usages
        private final Rating rating;
        2 usages
        private final String itemName;
        2 usages
        private final String imageName;
        2 usages
        private final Double silverPercent;
        2 usages
        private final Double goldPercent;

        1 usage
        public GameArtifactBuilder(GameArtifactCreateRequest createRequest) {
            this.rating = createRequest.getRating();
            this.itemName = createRequest.getItemName();
            this.imageName = createRequest.getImageName();
            this.silverPercent = createRequest.getSilverPercent();
            this.goldPercent = createRequest.getGoldPercent();
        }

        @Override
        public GameArtifact build() { return new GameArtifact( builder: this); }
    }
}
```

GameArtifact Entity (2)

```
getMoney() {
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
    this.$store.dispatch(this.$gameApiConstants.DO_SHOW_ME_THE_MONEY)
        .then((res) => {
            this.payCount = res.data.data.payCount
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
        .catch((err) => {
            this.$toast.error(err.response.data.msg)
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
},
chooseBoxSilver() {
    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
    this.$store.dispatch(this.$gameApiConstants.DO_CHOOSE_BOX_SILVER)
        .then((res) => {
            this.payCount = res.data.data.moneyResponse.payCount
            this.artifactList = res.data.data.useArtifactItems
            this.currentChooseArtifact = res.data.data.currentItem
            this.artifactDialogVisible = true
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
        .catch((err) => {
            this.$toast.error(err.response.data.msg)
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
        })
},
```

```
        chooseBoxGold() {
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
            this.$store.dispatch(this.$gameApiConstants.DO_CHOOSE_BOX_GOLD)
                .then((res) => {
                    this.payCount = res.data.data.moneyResponse.payCount
                    this.artifactList = res.data.data.useArtifactItems
                    this.currentChooseArtifact = res.data.data.currentItem
                    this.artifactDialogVisible = true
                    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
                })
                .catch((err) => {
                    this.$toast.error(err.response.data.msg)
                    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
                })
        },
        gameReset() {
            this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: true)
            this.$store.dispatch(this.$gameApiConstants.DO_GAME_RESET)
                .then((res) => {
                    this.getFirstData()
                })
                .catch((err) => {
                    this.$toast.error(err.response.data.msg)
                    this.$store.commit(this.$customLoadingConstants.FETCH_LOADING_SHOW, payload: false)
                })
        }
    },
    created() {
        this.getFirstData()
    },
    mounted() {
        this.$store.commit(this.$menuConstants.FETCH_SELECTED_MENU, payload: 'DASH_BOARD')
    },
}
</script>
```

Money Entity

```java
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class Money {
    no usages
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    4 usages
    @ApiModelProperty(notes = "돈 수량")
    @Column(nullable = false)
    private Long payCount;

    1 usage
    public void resetMoney() {
        this.payCount = 0L;
    }

    1 usage
    public void plusMoney() {
        this.payCount += 10;
    }
    1 usage
    public void minusMoney(long money) { this.payCount -= money; }

    1 usage
    private Money(MoneyBuilder builder) { this.payCount = builder.payCount; }

    2 usages
    public static class MoneyBuilder implements CommonModelBuilder<Money> {
        2 usages
        private final Long payCount;

        1 usage
        public MoneyBuilder() { this.payCount = 0L; }

        @Override
        public Money build() { return new Money( builder: this); }
    }

}
```

UseArtifact Entity

```java
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class UseArtifact {
    no usages
    @ApiModelProperty(notes = "시퀀스")
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    1 usage
    @ApiModelProperty(notes = "게임유물")
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "gameArtifactId", nullable = false)
    private GameArtifact gameArtifact;

    3 usages
    @ApiModelProperty(notes = "수량")
    @Column(nullable = false)
    private Integer artifactCount;

    1 usage
    public void resetCount() { this.artifactCount = 0; }

    1 usage
    public void putCountPlus() { this.artifactCount += 1; }
```

```java
1 usage
public void resetCount() { this.artifactCount = 0; }

1 usage
public void putCountPlus() { this.artifactCount += 1; }

1 usage
private UseArtifact(UseArtifactBuilder builder) {
    this.gameArtifact = builder.gameArtifact;
    this.artifactCount = builder.artifactCount;
}

2 usages
public static class UseArtifactBuilder implements CommonModelBuilder<UseArtifact> {
    2 usages
    private final GameArtifact gameArtifact;
    2 usages
    private final Integer artifactCount;

    1 usage
    public UseArtifactBuilder(GameArtifact gameArtifact) {
        this.gameArtifact = gameArtifact;
        this.artifactCount = 0;
    }

    @Override
    public UseArtifact build() { return new UseArtifact( builder: this); }
}
```

ChooseArtifact Controller

```java
@Api(tags = "보석 채굴")
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/choose-artifact")
public class ChooseArtifactController {
    2 usages
    private final ArtifactChooseService artifactChooseService;

    no usages
    @ApiOperation(value = "일반 채굴")
    @PostMapping("/silver")
    public SingleResult<ChooseArtifactResponse> chooseSilverBox() {
        return ResponseService.getSingleResult(artifactChooseService.getResult( isGoldBox: false));
    }

    no usages
    @ApiOperation(value = "특별 채굴")
    @PostMapping("/gold")
    public SingleResult<ChooseArtifactResponse> chooseGoldBox() {
        return ResponseService.getSingleResult(artifactChooseService.getResult( isGoldBox: true));
    }
}
```

GameData Controller

```java
@Api(tags = "게임 데이터 관리")
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/first-connect")
public class GameDataController {
    2 usages
    private final GameDataService gameDataService;

    no usages
    @ApiOperation(value = "첫 접속시 현황 데이터")
    @GetMapping("/init")
    public SingleResult<FirstConnectDataResponse> getFirstData() {
        return ResponseService.getSingleResult(gameDataService.getFirstData());
    }

    no usages
    @ApiOperation(value = "게임 정보 리셋")
    @DeleteMapping("/reset")
    public CommonResult resetGameData() {
        gameDataService.gameReset();

        return ResponseService.getSuccessResult();
    }
}
```

Money Controller

```java
@Api(tags = "돈 관리")
@RestController
@RequiredArgsConstructor
@RequestMapping("/v1/money")
public class MoneyController {
    1 usage
    private final MoneyService moneyService;

    no usages
    @ApiOperation(value = "돈 증가(광석캐기)")
    @PutMapping("/plus")
    public SingleResult<MoneyResponse> plusMoney() {
        return ResponseService.getSingleResult(moneyService.putMoneyPlus());
    }
}
```

ArtifactChoose Service (1)

```java
@Service
@RequiredArgsConstructor
public class ArtifactChooseService {
    3 usages
    private final MoneyRepository moneyRepository;
    1 usage
    private final GameArtifactRepository gameArtifactRepository;
    3 usages
    private final UseArtifactRepository useArtifactRepository;

    3 usages
    long choosePay = 0L;

    1 usage
    private void init(boolean isGoldBox) {
        this.choosePay = isGoldBox ? 1000L : 100L; // 뽑기비용... 금박스 여는거면 5000원, 은박스 여는거면 500원
    }
```

ArtifactChoose Service (2)

```java
public ChooseArtifactResponse getResult(boolean isGoldBox) {
    this.init(isGoldBox); // 초기화 시킨다.

    boolean isEnoughMoney = this.isStartChooseByMoneyCheck(isGoldBox); // 돈 충분한지 확인한다.

    if (!isEnoughMoney) throw new CNotEnoughMoneyException(); // 돈이 충분하지 않으면 뽑기 진행 불가능.. 예외처리 exception은 알아서 추가하기

    Money money = moneyRepository.findById(1L).orElseThrow(CMissingDataException::new);
    money.minusMoney(this.choosePay); // 뽑기 비용만큼 돈 소모시키기
    moneyRepository.save(money); // 돈 마이너스 시킨거 저장해서 확정하기

    long artifactResultId = this.getChooseArtifact(isGoldBox); // 랜덤으로 뽑은 유물 id 받아오기

    boolean isNewArtifact = false; // 새로 뽑은 유물인지 검사하기 위해 변수 추가하는데 기본으로 아니라고(false) 함.
    Optional<UseArtifact> useArtifact = useArtifactRepository.findByGameArtifact_Id(artifactResultId); // 뽑힌 유물 id와 일치하는 보유 유물 데이터를 가져온다.
    if (useArtifact.isEmpty()) throw new CMissingDataException(); // 만약 보유유물 데이터가 없으면 게임진행이 안되므로 오류메세지 출력..
    UseArtifact useArtifactData = useArtifact.get(); // 명확성을 위해 UseArtifact 모양으로 상자 하나 만들어서 거기다가 원본 옮겨놓기

    if (useArtifactData.getArtifactCount() == 0) isNewArtifact = true; // 만약 원본데이터에서 보유유물 갯수가 0개라면 이건 없던걸 뽑은거니까 isNewArtifact 를 true

    useArtifactData.putCountPlus(); // 보유유물 갯수 +1 해주기
    useArtifactRepository.save(useArtifactData); // 갯수 + 1 해준거.. (수정된거) 반영 확정하기

    // 위에서 처리는 다 되었고 이제 정보 돌려줄 준비 하기
    ChooseArtifactResponse result = new ChooseArtifactResponse();
    result.setMoneyResponse(new MoneyResponse.MoneyResponseBuilder(money).build());
    result.setCurrentItem(new ChooseArtifactCurrentItem.ChooseArtifactCurrentItemBuilder(useArtifactData.getGameArtifact(), isNewArtifact).build());
    result.setUseArtifactItems(this.getMyArtifacts());

    return result;
}
```

ArtifactChoose Service (3)

```java
/**
 * 내가 보유한 유물 컬렉션을 가져온다.
 *
 * @return 보유한 유물 컬렉션
 */
1 usage
public List<UseArtifactItem> getMyArtifacts() {
    List<UseArtifact> originList = useArtifactRepository.findAllByIdGreaterThanEqualOrderByIdAsc(1L); // 내가 가지고

    List<UseArtifactItem> result = new LinkedList<>(); // 결과값을 담을 빈 리스트를 생성한다.

    originList.forEach(item -> result.add(new UseArtifactItem.UseArtifactItemBuilder(item).build()));

    return result;
}


/**
 * 뽑기를 진행함에 있어 돈이 충분한지 확인해주는 메서드
 *
 * @param isGoldBox 금박스 뽑기 여부
 * @return true : 충분하다 / false : 불충분하다
 */
1 usage
private boolean isStartChooseByMoneyCheck(boolean isGoldBox) {
    Money money = moneyRepository.findById(1L).orElseThrow(CMissingDataException::new); // 돈 데이터 가져오기.. 얘는 무

    boolean result = false; // 기본으로 일단 막아두기...

    if (money.getPayCount() >= choosePay) result = true; // 내가 가진돈이 뽑기비용보다 많거나 같으면 체크 결과를 true로 바꾼다

    return result;
}
```

ArtifactChoose Service (4)

```
private long getChooseArtifact(boolean isGoldBox) {
    List<GameArtifact> artifacts = gameArtifactRepository.findAll();

    List<ChooseArtifactItem> percentBar = new LinkedList<>(); // 확률bar 결과 넣는 리스트

    double oldPercent = 0D; // 확률 누적값 담을 변수 생성
    for (GameArtifact artifact : artifacts) { // 유물 리스트에서 유물을 하나씩 던져주면서 반복시작한다.
        ChooseArtifactItem addItem = new ChooseArtifactItem(); // 확률bar를 채우기 위한 새 그릇을 만든다.
        addItem.setId(artifact.getId()); // 확률bar용 그릇에 유물id값을 넣는다.
        addItem.setPercentMin(oldPercent); // 확률bar용 그릇에 확률 시작값 세팅
        if (isGoldBox) { // 만약에 금상자를 여는것이라면
            addItem.setPercentMax(oldPercent + artifact.getGoldPercent()); // 확률bar용 그릇에 확률 종료값
        } else { // 그게 아니라면.. (= 은상자 여는거)
            addItem.setPercentMax(oldPercent + artifact.getSilverPercent()); // 확률bar용 그릇에 확률 종료값
        }

        percentBar.add(addItem); // 확률bar 결과 넣는 리스트에 위에서 만든 그릇.. 세팅 다 된 그릇 추가

        if (isGoldBox) {
            oldPercent += artifact.getGoldPercent(); // 확률 누적값에 확률 누적 (금 확률값)
        } else {
            oldPercent += artifact.getSilverPercent(); // 확률 누적값에 확률 누적 (은 확률값)
        }

    }

    double percentResult = Math.random() * 100; // 랜덤으로 0~100 사이 수 하나 뽑아오기

    long resultId = 0; // 뽑기 결과 유물id 담을 변수 만들기
    for (ChooseArtifactItem item : percentBar) { // 확률bar 안에 아이템 검사하면서 랜덤으로 뽑은 수치가 어느 구간
        if (percentResult >= item.getPercentMin() && percentResult <= item.getPercentMax()) { // 만약 랜
            resultId = item.getId(); // 뽑기 결과 유물id에 현재 구간의 id를 뽑아다가 넣고
            break; // 반복 종료하기
        }
    }

    return resultId;
}
```

GameData Service

```java
@Service
@RequiredArgsConstructor
public class GameDataService {
    3 usages
    private final MoneyRepository moneyRepository;
    3 usages
    private final UseArtifactRepository useArtifactRepository;

    /**
     * 게임 접속하면 맨 처음 받아올 데이터
     *
     * @return
     */
    1 usage
    public FirstConnectDataResponse getFirstData() {
        FirstConnectDataResponse result = new FirstConnectDataResponse(); // 결과값을 담을 빈

        Money money = moneyRepository.findById(1L).orElseThrow(CMissingDataException::new);
        result.setMoneyResponse(new MoneyResponse.MoneyResponseBuilder(money).build()); //
        result.setUseArtifactItems(this.getMyArtifacts()); // 보유 유물 컬렉션 넣기

        return result;
    }
```

```java
    /**
     * 게임 정보 리셋
     */
    1 usage
    public void gameReset() {
        Money money = moneyRepository.findById(1L).orElseThrow(CMissingDataException::new); // 돈 원본 데이터
        money.resetMoney(); // 돈 0원으로 교체
        moneyRepository.save(money); // 저장해서 확정

        List<UseArtifact> originList = useArtifactRepository.findAll(); // 내가 가지고있는 유물 리스트 전체를 불러온
        for (UseArtifact item : originList) { // 유물 리스트에서 유물 하나씩 던져주면서 반복시작
            item.resetCount(); // 유물 보유 수량 0으로 리셋하고
            useArtifactRepository.save(item); // 저장해서 확정
        }
    }

    /**
     * 내가 보유한 유물 컬렉션을 가져온다.
     *
     * @return 보유한 유물 컬렉션
     */
    1 usage
    public List<UseArtifactItem> getMyArtifacts() {
        List<UseArtifact> originList = useArtifactRepository.findAllByIdGreaterThanEqualOrderByIdAsc(1L); //

        List<UseArtifactItem> result = new LinkedList<>(); // 결과값을 담을 빈 리스트를 생성한다.

        originList.forEach(item -> result.add(new UseArtifactItem.UseArtifactItemBuilder(item).build()));

        return result;
    }
}
```

InitData Service (1)

```java
@Service
@RequiredArgsConstructor
public class InitDataService {
    2 usages
    private final MoneyRepository moneyRepository;
    3 usages
    private final GameArtifactRepository gameArtifactRepository;
    2 usages
    private final UseArtifactRepository useArtifactRepository;

    /**
     * 첫 돈을 세팅한다. (0원으로..)
     */
    1 usage
    public void setFirstMoney() {
        Optional<Money> originData = moneyRepository.findById(1L); // 1번 id를 가

        if (originData.isEmpty()) { // 만약 originData가 없으면....
            Money addData = new Money.MoneyBuilder().build(); // Money 기본값을 세
            moneyRepository.save(addData); // DB에 저장한다... 그러면 1번으로 0원짜리
        }
    }

    /**
     * 게임 유물 정보들을 세팅한다.
     */
```

InitData Service (2)

```java
public void setFirstArtifact() {
    List<GameArtifact> originList = gameArtifactRepository.findAll(); // 등록된 유물 리스트를 가져온다.

    if (originList.size() == 0) { // 등록된 유물 갯수가 하나도 없으면...
        List<GameArtifactCreateRequest> result = new LinkedList<>(); // 보석 정보들이 들어갈 빈 리스트 생성
        GameArtifactCreateRequest request1 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.NORMAL, itemName: "normal1", imageName: "normal1.jpg", silverPercent: 12D, goldPercent: 0D).build();
        result.add(request1);
        GameArtifactCreateRequest request2 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.NORMAL, itemName: "normal2", imageName: "normal2.jpg", silverPercent: 12D, goldPercent: 0D).build();
        result.add(request2);
        GameArtifactCreateRequest request3 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.NORMAL, itemName: "normal3", imageName: "normal3.jpg", silverPercent: 12D, goldPercent: 0D).build();
        result.add(request3);
        GameArtifactCreateRequest request4 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.NORMAL, itemName: "normal4", imageName: "normal4.jpg", silverPercent: 12D, goldPercent: 0D).build();
        result.add(request4);
        GameArtifactCreateRequest request5 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.NORMAL, itemName: "normal5", imageName: "normal5.jpg", silverPercent: 12D, goldPercent: 0D).build();
        result.add(request5);
        GameArtifactCreateRequest request6 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.NORMAL, itemName: "normal6", imageName: "normal6.jpg", silverPercent: 12D, goldPercent: 0D).build();
        result.add(request6);
        GameArtifactCreateRequest request7 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.RARE, itemName: "rare1", imageName: "rare1.jpg", silverPercent: 6D, goldPercent: 25D).build();
        result.add(request7);
        GameArtifactCreateRequest request8 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.RARE, itemName: "rare2", imageName: "rare2.jpg", silverPercent: 6D, goldPercent: 25D).build();
        result.add(request8);
        GameArtifactCreateRequest request9 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.RARE, itemName: "rare3", imageName: "rare3.jpg", silverPercent: 6D, goldPercent: 25D).build();
        result.add(request9);
        GameArtifactCreateRequest request10 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.UNIQUE, itemName: "emerald", imageName: "emerald.jpg", silverPercent: 4.5D, goldPercent: 10D).build();
        result.add(request10);
        GameArtifactCreateRequest request11 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.UNIQUE, itemName: "sapphire", imageName: "sapphire.jpg", silverPercent: 4.5D, goldPercent: 10D).build();
        result.add(request11);
        GameArtifactCreateRequest request12 = new GameArtifactCreateRequest.GameArtifactCreateRequestBuilder
                (Rating.LEGENDARY, itemName: "diamond", imageName: "diamond.jpg", silverPercent: 1D, goldPercent: 5D).build();
        result.add(request12);

        result.forEach(item -> {
            GameArtifact addData = new GameArtifact.GameArtifactBuilder(item).build();
            gameArtifactRepository.save(addData);
        });
    }
}

1 usage
public void setFirstUseArtifact() {
    List<UseArtifact> useArtifacts = useArtifactRepository.findAll(); // db에서 등록되어있는 모든 보유유물들을 가져온다.

    if (useArtifacts.size() == 0) { // 보유유물 데이터가 하나도 없으면..
        List<GameArtifact> gameArtifacts = gameArtifactRepository.findAll(); // 등록된 유물 리스트를 불러온다.

        gameArtifacts.forEach(item -> { // 유물리스트에서 유물을 하나씩 던져주면서
            UseArtifact addData = new UseArtifact.UseArtifactBuilder(item).build(); // 보유유물 임시 데이터로 만들고
            useArtifactRepository.save(addData); // DB에 저장한다.
        });
```

Money Service

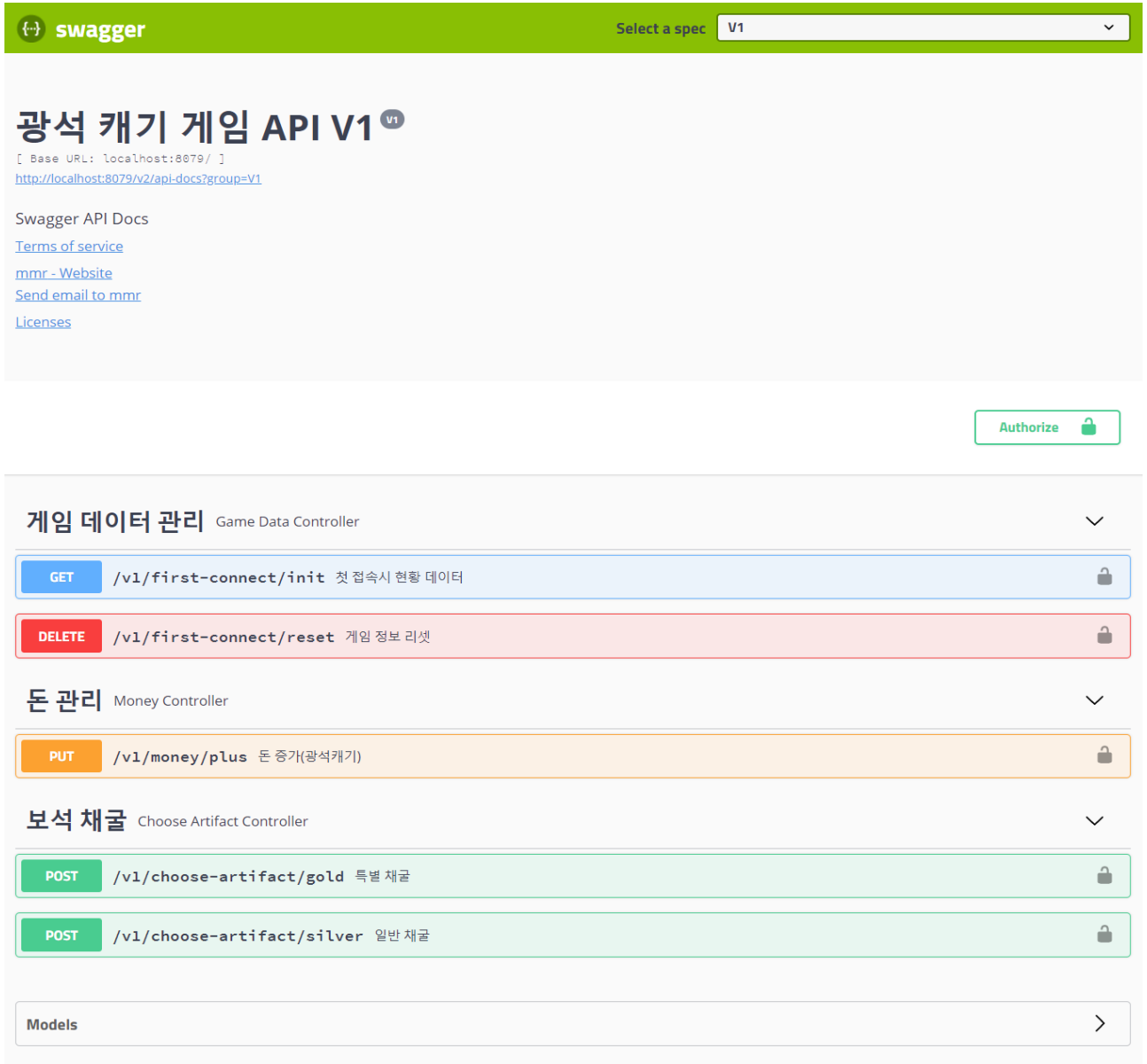
```java
@Service
@RequiredArgsConstructor
public class MoneyService {
    2 usages
    private final MoneyRepository moneyRepository;

    1 usage
    public MoneyResponse putMoneyPlus() {
        Money money = moneyRepository.findById(1L).orElseThrow(CMissingDataException::new);
        money.plusMoney(); // 엔티티 안쪽에 put기능을 하는 메서드를 이용해서 500원 증가시키기

        Money result = moneyRepository.save(money); // 저장하고 저장한 결과를 받아오기

        return new MoneyResponse.MoneyResponseBuilder(result).build(); // MoneyResponse를 주
    }
```

Swagger-ui 화면



Part 5

프론트 시연

광산

현재 보유한 돈 : 0원

← 모인 돈

광석 캐기! (10원)

뽑기

일반 채굴 (100원)

특별 채굴 (1,000원)

실제 구현 동영상 보기 (클릭)

보석 컬렉션

보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개

보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개 | 보석정보없음 -개

게임리셋

The End