



네루룽 봇

관리, 노래, 게임 등의 통합 디스코드 봇

[소개 페이지](#)

[BOT 초대링크](#)

[서포트 디스코드](#)

01 요약

서버사이드 & 백엔드



Discord.js 기반의 봇

Express

소개 홈페이지 구축용 프레임워크



HTTPS 인증서 적용으로 보안강화

인프라 & 배포



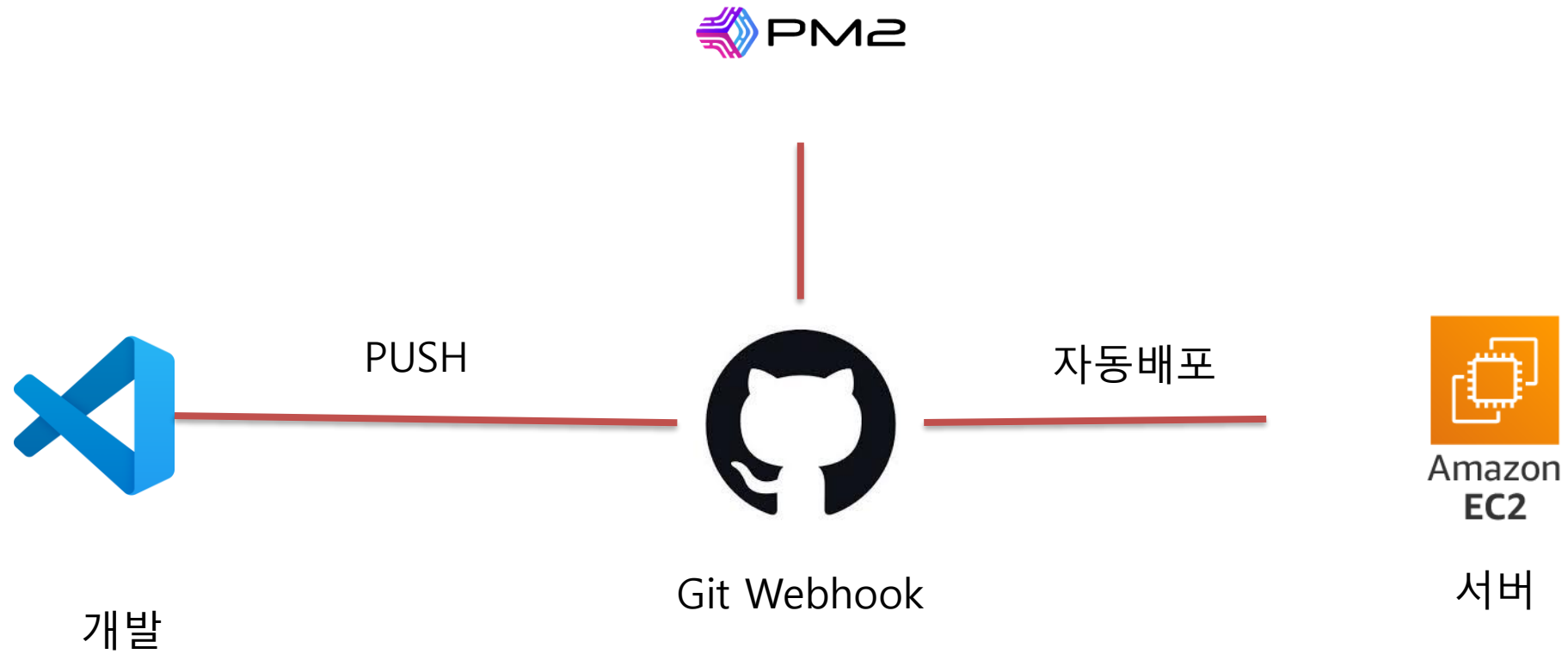
Amazon
EC2

호스팅 (t2.micro)

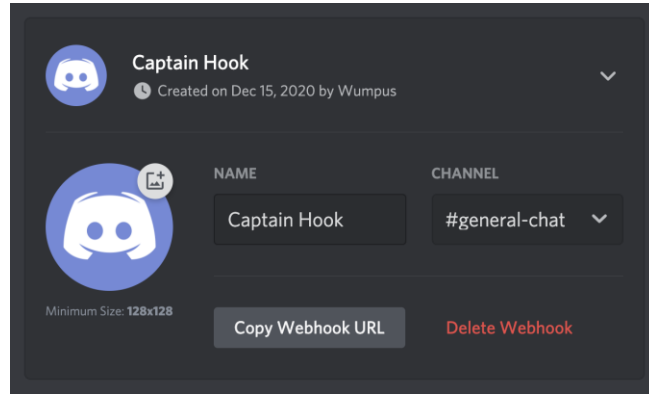


프로세스 매니저
무중단 실행 및 배포시 재시작 관리

자동화 배포





시스템 모니터링





watchBotStatus.js가 정각마다 Discord Hook 으로 보고


 ServerStatus **앱** 오후 5:00


 서버 상태 자동 보고


 CPU 부하
0.13

 메모리
312MB / 957MB


 Uptime
2512분

 Webhook 응답
● 정상 (200 OK)

 도메인 응답
● 정상 (200 OK)

 디스크 사용량

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/root	6.8G	4.5G	2.3G	67%	/
/dev/xvda16	881M	137M	683M	17%	/boot
/dev/xvda15	105M	6.1M	99M	6%	/boot/efi

 PM2 상태

● nerubot	online	100.4mb	0.5%
● server-intro	online	45.4mb	0.3%
● webhook	online	8.3mb	0%
● watchBotStatus	online	67.2mb	0.3%

보고 시각: 2025. 4. 18. 오후 5:00:00 • 오늘 오후 5:00

봇의 기능

DISCORDJS

관리, 은행, RPG게임
Discord.js 기반



bser-open-api-handler-release

Endpoint <https://open-api.bser.io>

Description Eternal Return Open API

Unsubscribe

각종 api 사용으로 전적검색과
Chat GPT 이용등



Yt-dlp + Ffmpeg 기반 유튜브 노래 재생기능

봇의 기능

네루릉 앱 오전 2:15



⚠️ gonicdeath 님에게 경고가 부여되었습니다.

사유
사유 없음
경고 누적
1회

경고를 받을 경우 추가 제재가 있을 수 있습니다.

겜스갯흥엔진 오전 2:16
!계좌

@겜스갯흥엔진 !계좌

네루릉 앱 오전 2:16
💰 현재 잔액: 1200 코인



겜스갯흥엔진 오전 2:16
!노래 안녕하세요반가워요

네루릉 앱 오전 2:16
🎵 아따맘마 오프닝 '잘 있어요' - 리메이크 [4K] 을(를) 재생합니다!



@겜스갯흥엔진 !캐릭터정보

네루릉 앱 오전 2:17

GGE
LV.1 전사

일기당천

STR	8 (6 + 2)
DEX	4
INT	4
LUK	4

HP	MP	EXP
150	50	0 / 15



관리, 노래, 은행, RPG게임

02 구현

명령어 트리

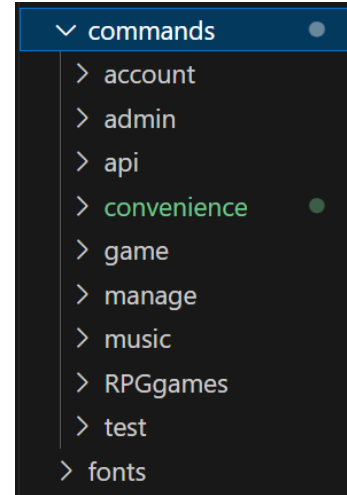
```
const getCommandFiles = (dir) => {
  let files = [];
  const items = fs.readdirSync(dir, { withFileTypes: true });

  for (const item of items) {
    const fullPath = path.join(dir, item.name);
    if (item.isDirectory()) {
      files = [...files, ...getCommandFiles(fullPath)];
    } else if (item.name.endsWith(".js")) {
      files.push(fullPath);
    }
  }
  return files;
};

const commandFiles = getCommandFiles("./commands");
for (const file of commandFiles) {
  const command = require(path.resolve(file));

  const relativePath = path.relative("./commands", file);
  const category = path.dirname(relativePath).replace(/\\/g, "/");

  command.category = category || "기타";
  client.commands.set(command.name, command);
}
```



봇이 가동될 때 index.js 에서
commends 폴더를
재귀적으로 탐색하여
폴더별로 카테고리 분류

명령어 추가가 자유롭도록
확장성을 생각한 설계

카테고리로 분류된 명령어들은
서버 설정에 따라 사용 가능 / 불가 여부가 정해짐
카테고리는 이후 !도움말 명령어에서 사용가능한 명령어
목록을 보여줄때 다시 사용됨

명령어 템플릿

```
module.exports = {
  name: "명령어키워드 ",
  example: "!도움말 명령어에 붙는 사용예시 ",
  description: "!도움말 명령어에 붙는 기능소개",

  async execute(message, args) {
    try {
      const user = message.author;
      await message.reply("✅ 결과 보고.");

    } catch (error) {
      message.reply(`⚠️ 오류 발생: ${error.message}`);
    }
  },
};
```

이 템플릿에 맞춰 명령어를 작성하면
다른 작업이 필요없이 바로 prefix를 붙인 명령어키워드로
명령어를 작동시킬수 있음

DB FREE 설계

```
async getBalance(message, user){
  const accountMessage = await utility.findUserMessageByChannelName(message, "장부", user);
  if (!accountMessage) throw new Error(`${user.displayName}의 계좌가 없습니다!`);
  const match = accountMessage.content.match(/(\d+) 코인/);
  if (!match) throw new Error("기존 계좌의 저장형식이 잘못되었습니다. 관리자에게 문의하세요.");

  const balance = parseInt(match[1]);
  return balance;
},
```

장부



인벤토리



네루롱 앱 오전 1:25

367958092270534657: 1100 코인 (수정됨)

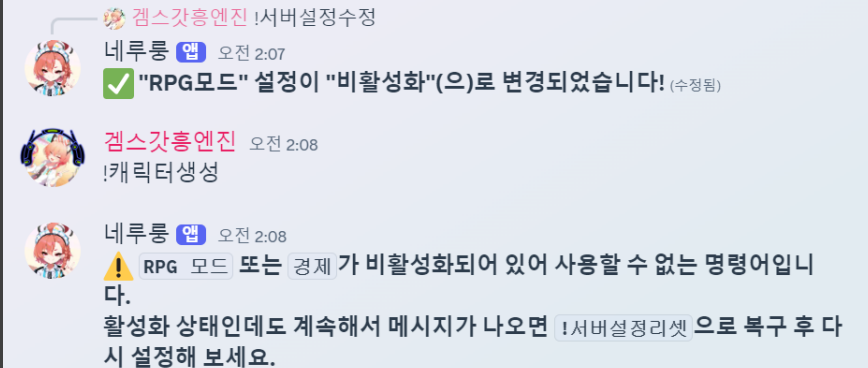
네루롱 봇은 외부 DB를 사용하지 않고
디스코드 채널에 정보를 읽고 씀
DB 호스팅 사용료가 부담되는 부분도 있었으나
1인 개발이라는 특성상 혼자서 RPG 아이템 증발 / 캐릭터 증발 등
버그와 관련 없는 각종 서비스 문의 까지 받게 되면 너무 바빠질것을 우려하여
각 서버장들이 DB를 직접 보고 처리 할 수 있도록 함

명령어 모음 ON/OFF

```
const isEconomyEnabled = settings["경제"] === "활성화";
const isRPGEEnabled = isEconomyEnabled && settings["RPG모드"] === "활성화";


const restrictionMessages = {
  economy: "⚠️ **`경제` 기능이 비활성화되어 있어 사용할 수 없는 명령어입니다.\n활성화 상태인데도 계속해서 메시지가 나오면 `!서버설정리셋`으로 복구 후 다시 설정해 보세요.**",
  RPGgames: "⚠️ **`RPG 모드` 또는 `경제`가 비활성화되어 있어 사용할 수 없는 명령어입니다.\n활성화 상태인데도 계속해서 메시지가 나오면 `!서버설정리셋`으로 복구 후 다시 설정해 보세요.**"
};


if (!isEconomyEnabled && command.category.startsWith("economy")) {
  return message.channel.send(restrictionMessages.economy);
}
if (!isRPGEEnabled && command.category.startsWith("RPGgames")) {
  return message.channel.send(restrictionMessages.RPGgames);
}
```





네루통 봇은 크게 관리 / 노래 / 경제 / RPG게임 4개의 카테고리를 가지고 있는데 더 마음에드는 타 봇과 명령어가 겹치거나 RPG 기능 없이 담백하게 노래 재생 기능만 사용하고싶다
이런 경우를 위해 서버 설정에서 해당 카테고리를 비활성화 할수있게 함
서버 설정은 앞서 소개 한 것처럼 "관리기록" 채널에 메시지로써 저장되어있음


서버초기화


 **겜스갯흥엔진** 오전 2:09
!안녕


 @겜스갯흥엔진 !안녕

 **네루롱 앱** 오전 2:09
!관리자는 !서버초기화 명령으로 봇을 사용 전에 기본 설정을 완료해주세요!

 **겜스갯흥엔진** 오전 2:09
!서버초기화

 @겜스갯흥엔진 !서버초기화

 **네루롱 앱** 오전 2:09
DM으로 서버 초기화 메시지가 발송되었습니다.


 **네루롱 앱** 오전 2:09

✅ **네루롱 봇 초기화 안내**
네루롱 봇이 정상 동작하기 위한 채널들을 생성합니다.


관리기록 - 서버 설정과 관리 기록 저장
장부 - 서버원들의 재화 수량 저장
인벤토리 - 서버원들의 RPG 정보

해당 채널들은 관리자와 봇만 접근할 수 있습니다.
이 채널의 메시지를 삭제하거나
초기화 진행을 위해 아래 버튼을 클릭

확인했습니다.

 **네루롱 앱** 오전 2:10

경제 : 활성화
RPG모드 : 활성화
계좌계설 축하금 : 100
송금 수수료 : 5
최대 경고 횟수 : 3 (수정됨)

 **네루롱 앱** 오전 2:09
(수정됨)

✅ **네루롱 봇 서포트 디스코드 서버의 초기화가 완료되었습니다.**

!서버설정 - 해당 명령어로 설정 확인
!서버설정수정 - 설정을 재설정합니다.

경제, 게임 기능을 원치 않으시면 !서버설정수정 에서 해당 명령어 단락을 비활성화 할 수 있습니다.
!도움말 에서 자세한 명령어 목록을 확인 하실수 있습니다.

네루롱 봇을 이용해주셔서 감사합니다!

앞서 소개 한것처럼 서버 설정이 local에 저장되기 때문에 저장공간이 만들어져야함
봇을 처음 초대하게 되면 서버초기화를 요구하고 초기화에 동의하면 채널, 서버설정이 생성됨

관리명령어

🔧 관리자 명령어

관리자 전용 명령어 목록입니다.

봇에게 관리자 권한이 없다면 채널/유저/역할 관리 명령어 사용이 제한됩니다.

🔧 관리자 명령어

!아이템지급 @유저 <아이템ID> <수량> - RPG전용 특정 유저에게 아이템을 지급합니다.

!입금 @유저 <금액> - 관리자가 특정 유저의 계좌에 돈을 추가합니다.

!공지 <내용> - 관리자 권한으로 공지를 보냅니다.

!출금 @유저 <금액> - 관리자가 특정 유저의 계좌에서 돈을 차감합니다.

채널 관리

!청소 <갯수> - 지정된 개수만큼 메시지를 삭제합니다.

!채널잠금 - 현재 채널을 잠급니다. (일반 사용자의 메시지 전송 제한)

!슬로우모드 <시간> - 채널에 슬로우 모드를 설정합니다.

!채널잠금해제 - 현재 채널의 잠금을 해제합니다.

유저 관리

!차단 @유저 <사유> - 서버에서 특정 유저를 차단합니다.

!차단목록 - 서버에서 차단된 유저들의 목록과 사유를 보여줍니다.

!추방 @유저 [사유] - 지정된 유저를 서버에서 추방합니다.

!뮤트 @유저 <시간 (예: 10s, 5m, 1h)> - 지정된 유저를 일정 시간 동안 뮤트합니다.

!차단해제 <유저ID> - 서버에서 특정 유저의 차단을 해제합니다.

!뮤트해제 @유저 - 지정된 유저의 뮤트를 해제합니다.

!경고삭제 @유저 [사유] - 특정 유저의 경고를 제거합니다.

!경고 @유저 <사유> - 특정 유저에게 경고를 부여합니다.

역할 관리

!역할 @유저 <역할이름> - 특정 유저에게 역할을 부여합니다.

!역할생성 <역할이름> <색상(선택)> - 서버에 새 역할을 생성합니다.

!역할삭제 <역할이름> - 서버에서 특정 역할을 삭제합니다.

!역할회수 @유저 <역할이름> - 특정 유저에게서 역할을 제거합니다.

!역할목록 - 서버의 모든 역할 목록을 표시합니다.

서버 설정

!서버설정 - 서버 설정을 확인합니다.

!서버초기화 - 서버의 기본 설정을 초기화합니다.

!서버설정리셋 - 서버 설정을 기본값으로 초기화합니다.

!서버설정수정 - 서버 설정을 수정합니다.

```
const reason = args.slice(1).join(" ") || "사유 없음";
try {
  await target.kick(`명령자: ${message.author.tag}, 사유: ${reason}`);
  return message.reply(`📢 **${target.user.tag}** 님이 서버에서 추방되었습니다.\n📢 사유: ${reason}`);
} catch (error) {
  console.error("❌ 추방 실패:", error);
  return message.reply("⚠️ 유저를 추방하는 데 실패했습니다.");
}
```

디스코드 채널의
유저/채널/역할을 관리할수
있는 명령어 모음으로

서버 설정을 다루는 부분을
제외한 대부분 discord.js
api가 제공하는 기능들의
간단한 활용

노래 명령어

```
const process = ytdlp.exec(song.url, {
  output: "-",
  format: "bestaudio",
  noPart: true,
  noCacheDir: true,
  bufferSize: "16K",
  socketTimeout: "30",
  quiet: false,
  verbose: true,
  youtubeSkipDashManifest: true,
  cookies: "cookies.txt",
});

const resource = createAudioResource(process.stdout);
if (!resource) {
  console.warn("⚠️ 유효하지 않은 오디오 리소스.
재생이 중단됩니다.");
  guildQueue.songs.shift();
  guildQueue.playing = false;
  return;
}
guildQueue.player.play(resource);
connection.subscribe(guildQueue.player);

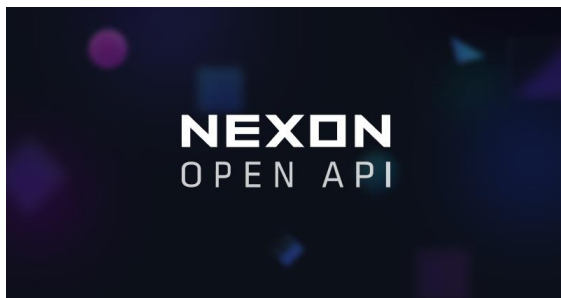
guildQueue.startTime = Date.now();
song.startTime = Date.now();

if (textChannel) {
  textChannel.send({ files: [attachment] });
  textChannel.send(`🎵 **${song.title}** 을(를)
재생합니다!`);
}
```



Yt-dly + Ffmpeg 조합으로
디스코드 음성 채널에서 노래 기능
사용가능

API



bser-open-api-handler-release

Endpoint <https://open-api.bser.io>

Description Eternal Return Open API

Unsubscribe



한동그라미



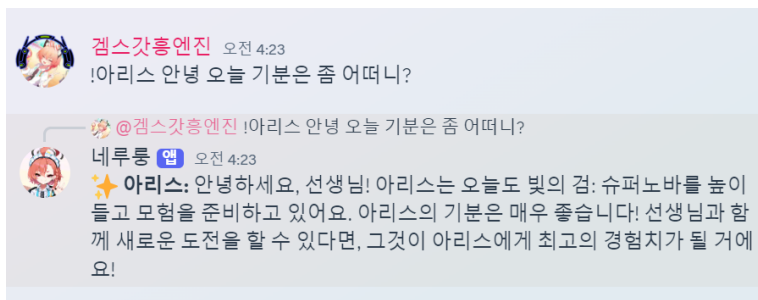
이터널리턴
9416 RP
5위 (상위 0.00%)

평균 TK: 11.11
평균 길: 5.08
평균 순위: 3.92
게임 수: 347
승률: 20.46%

각종 외부 API를 연결하여 봇에서 작동


NEXON OPEN API 중 메이플스토리 API를 사용해
유저의 메이플 캐릭터 정보를 가져와
Canvas에서 조립하여 보여줌

Bser-open-api 를 사용해
이터널리턴 유저의 최신 랭크게임
지표를 보여줌




OpenAI API를 사용해 Chat-GPT3.5o 를 연결해
아리스 컨셉의 대화 서비스를 제공함

RPG게임

네루릉  오전 4:29
전투를 시작합니다! (수정됨)

파란 달팽이



LV. 2
hp
144

| 전투로그 |

- ▶38◀ GGE의 "파워스트라이크"!
- ▶12◀ 파란 달팽이의 공격
- ▶35◀ GGE의 "분노의 강타"!
- ▶17◀ 파란 달팽이의 공격
- ▶37◀ GGE의 "분노의 강타"!

GGE
[HP] ●●●●○○○○○○ 74 / 150
[MP] ●●●●○○○○○○ 20 / 50

파란 달팽이
[HP] ○○○○○○○○○○○ 0 / 144 (수정됨)

파란 달팽이울(를) 처치했다!
전리품으로 파란달팽이 껍질울(를) 획득하고 10 exp, 6 코인을 얻었다!! (수정됨)

GGE의 경험치 (수정됨)
10 / 15 (66.67%)

GGE
LV.1 전사

일기당천

STR 8 (6 + 2)

DEX 4

INT 4

LUK 4

HP

150

MP

50

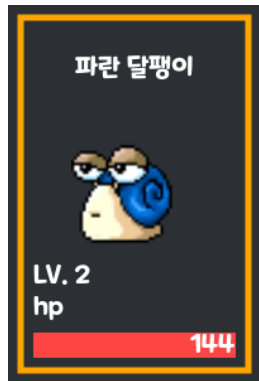
EXP

10 / 15



서버의 유저들은 자신의 캐릭터를 육성 할 수있음
전투를 통해 경험치를 얻고 레벨업으로 스탯을
분배하고
가진 아이템과 돈으로 서로 거래를 하고 PVP를
할수 있음

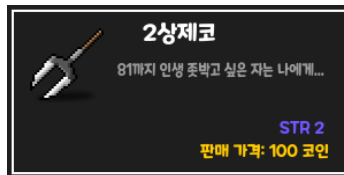
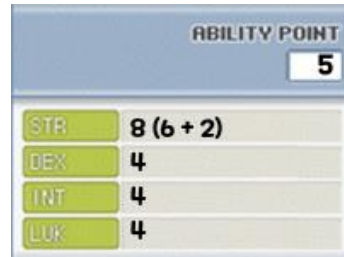
게임의 흐름



전투를
통해



레벨업과
전리품,
돈을 얻음



스탯을
분배하고
드랍된 더
좋은
아이템을
착용

Slot 0	Slot 1	Slot 2
Slot 3	Slot 4	Slot 5
Slot 6	Slot 7	Slot 8
Slot 9	Slot 10	Slot 11

교환신청
GGE → 테스트계정



2상제코 - 1개
100 코인

수락하려면 '수락'을 입력하세요.

아이템을
판매하거나
서로 교환



PVP 기능

전투 모듈 코드

```
async battle(message, character, monster) {
  const user = message.author;
  const username = message.author.displayName;

  let playerHP = character.maxHP;
  let playerMP = character.maxMP;
  let monsterHP = monster.hp;

  const battleLog = [];
  let turn = 1;

  const initialMessage = await message.channel.send("🗡️ 전투를 준비 중...");

  while (playerHP > 0 && monsterHP > 0) {
    if (turn % 2 === 1) {
      // 🎮 플레이어 턴
      const useSkill = playerMP > 0 && Math.random() < 0.7;
      const critical = Math.random() < 0.1;
      const baseDamage = Math.floor(await characterController.getCharacterStatAttackPower(message, user));
      let damage = useSkill ? Math.floor(skill(battleLog, character, username, baseDamage)) : baseDamage;

      if (!damage || isNaN(damage)) {
        battleLog.push(`${username}의 공격 빗나감`);
      } else {
        if (useSkill) playerMP -= 10;
        else battleLog.push(`**▶️ ${damage} ◀️** ${username}의 "일반공격!"`);

        if (critical) {
          damage *= 2;
          battleLog.push(`**${username}의 크리스탈!!**`, `**▶️▶️▶️ ${damage} ◀️◀️◀️**`);
        }
      }
    }
  }
}
```


거래 - 트랜잭션 생성

```
await tradingMessage.edit(`📄 트랜잭션을 생성하는 중입니다...`);
initialSenderBalance = await accountController.getBalance(message, sender);
initialReceiverBalance = await accountController.getBalance(message, receiver);
initialSenderInventory = await inventoryController.getInventory(message, sender);

await tradingMessage.edit(`💰 상대방의 잔액 확인 중...`);
if (initialReceiverBalance < price) throw new Error("❌ 거래 실패! 상대방의 잔액이 부족합니다.");

await tradingMessage.edit(`💸 금액을 전송 중...`);
await accountController.updateBalance(message, receiver, -price);
await accountController.updateBalance(message, sender, price);

await tradingMessage.edit(`📦 아이템을 전달 중...`);
let tradingAfterItemSlot = { id: "undefined", quantity: "undefined" };
if (item.quantity > quantity){
    tradingAfterItemSlot = { id: item.id, quantity: (item.quantity - quantity) };
}
await inventoryController.updateInventory(message, sender.id, { [slotIndex]: tradingAfterItemSlot });
await inventoryController.addItemToInventory(message, receiver.id, item.id, quantity);
receiverInventoryUpdated = true;

await tradingMessage.edit(`✅ 거래 완료! ${sender.displayName} → ${receiver.displayName}에게 **${itemData.name} x${quantity}** 전달됨!`);
```

DB를 사용하는 것이 아니기 때문에 비동기 환경에서 거래시 불공평한 상황이 발생 할 수있음
이를 유사 트랜잭션을 구현하여 원자성을 보장함

소개 페이지

```
if (!fs.existsSync(`${CERT_PATH}/privkey.pem`)
|| !fs.existsSync(`${CERT_PATH}/fullchain.pem`)) {
  console.error("인증서 파일이 존재하지 않습니다.");
  process.exit(1);
}

const options = {
  key: fs.readFileSync(`${CERT_PATH}/privkey.pem`),
  cert: fs.readFileSync(`${CERT_PATH}/fullchain.pem`),
};

app.use(helmet({
  contentSecurityPolicy: false
}));

app.use(express.static(path.join(__dirname)));

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'intro.html'));
});

https.createServer(options, app).listen(8080, () => {
  console.log(`HTTPS 서버 실행 중: https://${DOMAIN}`);
});
```

Express

Express 프레임워크를 통해
정적파일 html을 웹에 싸줌



Let's Encrypt에서 발급한 SSL
인증서를 이용해 HTTPS 서버를
구성하여 사용자 데이터를
암호화된 통신으로 보호합니다.
helmet 미들웨어를 활용해 기본
보안 HTTP 헤더(CSP, XSS 보호 등)
적용.

시스템 모니터링

```
cron.schedule("0 * * * *", async () => {
  try {
    const embed = await generateStatusEmbed();
    await webhook.send({ embeds: [embed.toJSON()] });
    console.log("🟢 정각 상태 보고 전송 완료");
  } catch (err) {
    console.error("❌ 상태 보고 실패:", err);
  }
}, {
  timezone: "Asia/Seoul"
});
setInterval(() => {
  exec(`pm2 show ${PROCESS_NAME}`, (err, stdout, stderr) => {
    const isErrored =
      err ||
      (stderr && stderr.toLowerCase().includes("not found")) ||
      stdout.toLowerCase().includes("errored") ||
      stdout.toLowerCase().includes("stopped");

    if (isErrored && !botIsDown) {
      botIsDown = true;
      webhook.send({
        content: `🚨 봇(${PROCESS_NAME})이 다운됐습니다!
확인해주세요.`
      }).catch(console.error);
    } else if (!isErrored) {
      botIsDown = false;
    }
  });
}, 60 * 1000);
```

디스코드 봇의 가용성과 안정성을
높이기 위해 pm2와 Webhook
기반의 모니터링 시스템을
구축했습니다.

매 정각마다 Discord Embed
형식으로 서버 상태 보고서 전송

pm2 show <process> 명령어를
통해 프로세스 상태 확인

Errored, Stopped, 또는 감지되지
않을 경우 Discord Webhook으로
경고 전송