2025.05

Capstone Design:
2nd Week Progress Overview

Web Development

이름: 이재민
학번: 202058096

2025.05 - 2주차 현황

1. 메인 페이지 구조 및 요청처리 설계

✅ 서버 환경

호스팅: **Cafe24 Tomcat (Java 11, Spring Boot** 기반)
다중 사이트 구성: 동일 서버 내 개별 프로젝트 공존
따라서 요청 처리가 중요.

왜?

클라이언트가 다른 **URL**을 검색하고 들어갈 경우, 원하는 사이트가 아닌 다른 사이트가 나올 가능성을 배제할 수 없음

✅ URI 및 요청 흐름 설계

사용자가 **https://plusb3b.kr**에 직접 접근할 경우:
예외 처리 차원에서 **/game-hub**로 **Redirect** 처리
/ 경로는 다른 프로젝트와 충돌 방지를 위해 비워두거나 안내페이지로 설정

/game-hub 요청 시:

Controller에서 **GET** 요청을 처리
메인 페이지인 **index.html**을 렌더링

2. 메인 페이지 구성요소 설계

✅ 메인 페이지 구성 목적

단일 **HTML** 파일에 모든 **UI** 요소를 포함하면 유지보수가 어려움
따라서, 코드 가독성과 재사용성 확보를 위해 페이지를 구성 요소별로 분리

왜?

만약 모든 요소들이 한 페이지 안에 들어가있다면, 몇 백줄씩 되는 코드를 검토하면서 유지보수하기가 힘들게 된다.
따라서 **header, nav,** 등으로 로 조각내어 개발 후, **index.html**에서 모두 **include**하여 보여지게 된다.

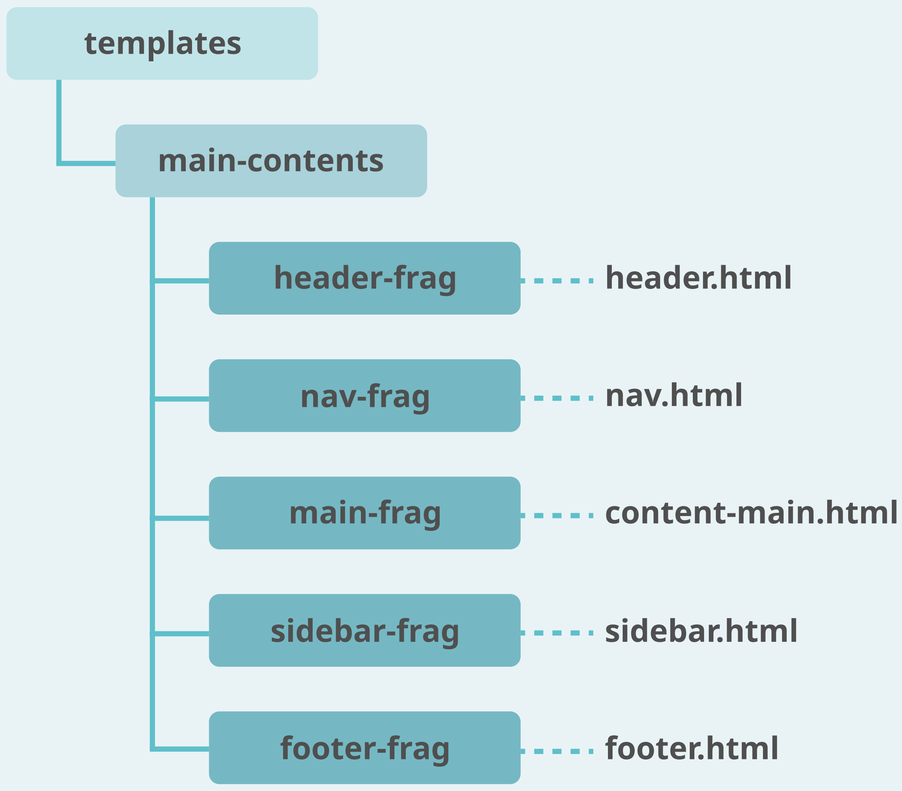
✅ 페이지 조각화 전략 (Fragmenting)

모듈화된 구성 요소

구성 요소	파일 경로	설명
header	templates/main-contents/header-frag/header.html	상단로고, 타이틀, 유저정보
nav	templates/main-contents/nav-frag/nav.html	상단 또는 좌측 메뉴 영역
content	templates/main-contents/main-frag/content-main.html	실제 콘텐츠 렌더링 영역
sidebar	templates/main-contents/sidebar-frag/side.html	보조 메뉴 또는 퀵 링크 영역
footer	templates/main-contents/footer-frag/footer.html	하단 저작권, 정보 표시 영역

최종적으로 **index.html**에서
th:replace 또는 **th:include** 방식으로 위 조각들을 불러와 화면 구성

2. 메인 페이지 구성요소 설계 - 파일구조 시각화



2025.05 - 2주차 현황

3. 버전 관리 전략 (VCS)

✅ 버전 관리 전략 (VCS)

GitHub을 통해 코드 변경 이력 관리

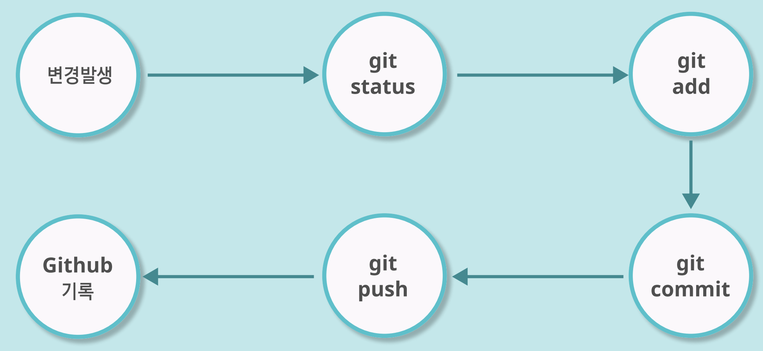
수정 사항 발생 시:

커밋 메시지에 날짜 / 버전 / 수정된 파일 및 내용 요약 명시

EX)

```
feat: 2025-05-07 / v1.1 / header.html 스타일 개선 및 반응형 처리
    주기적으로 push하여 안정된 버전을 저장
```

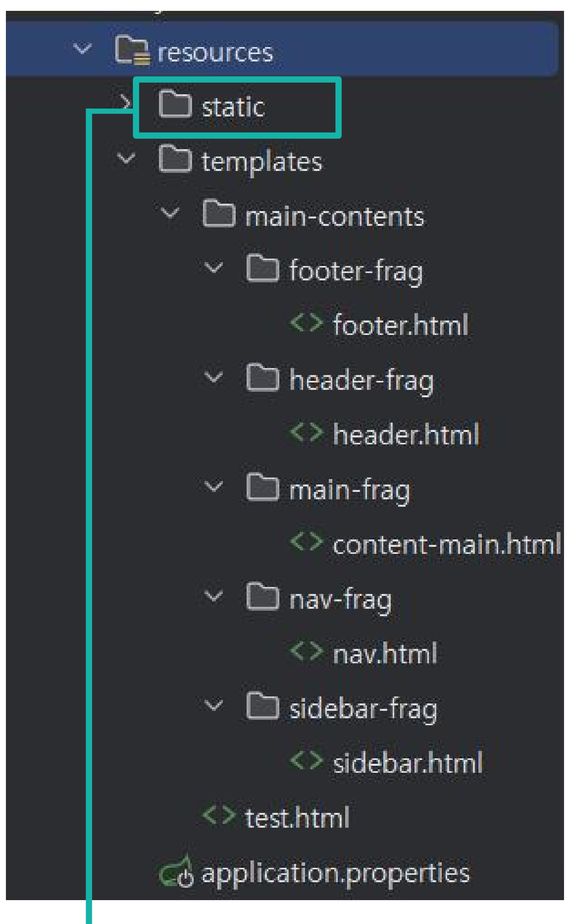
✅ 버전 관리 전략 (VCS) - 시각화



4. 향후 확장성 고려

✅ 향후 확장성 고려

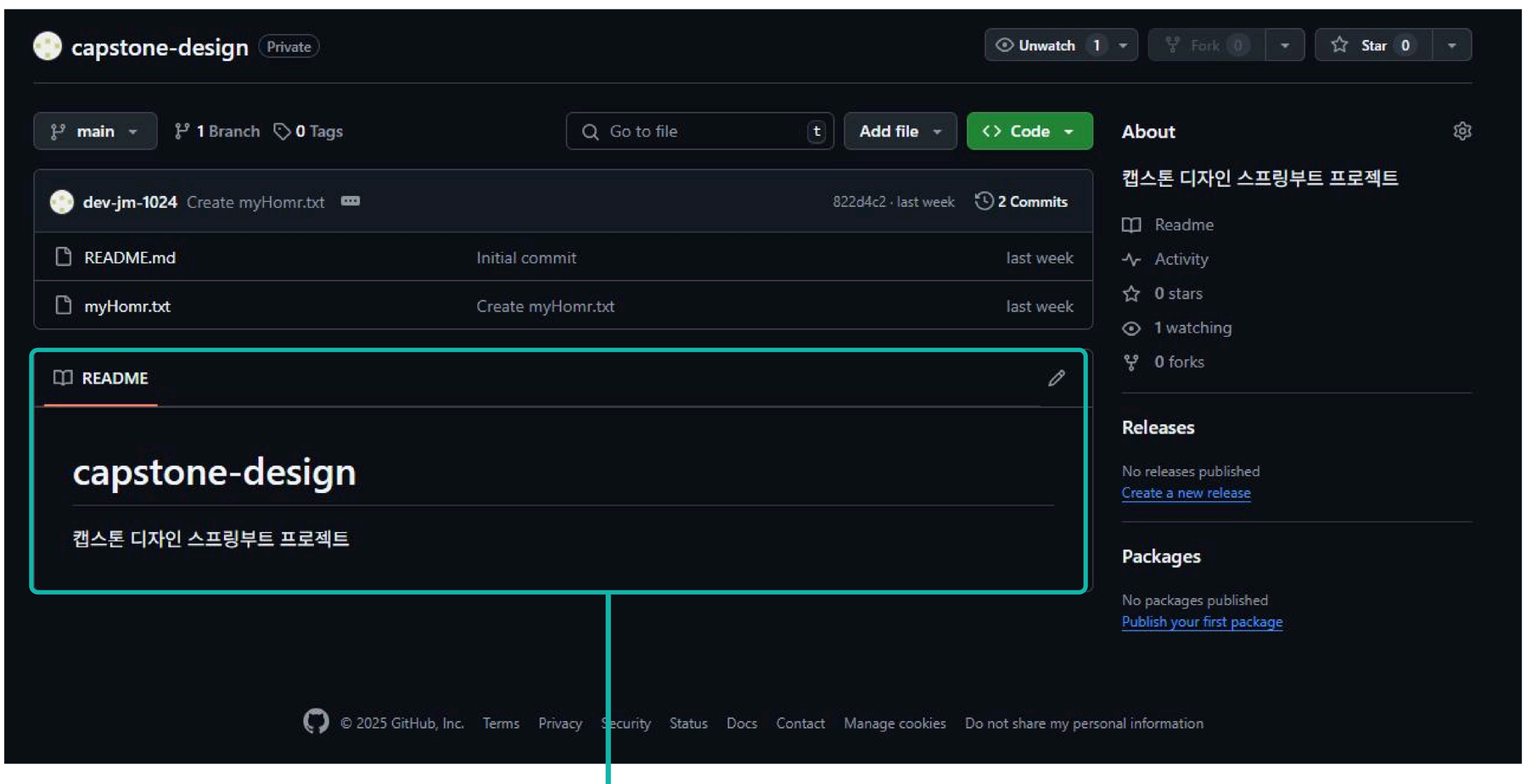
- 각 조각 파일은 다른 페이지에서도 재사용 가능하도록 독립성 확보
- 콘텐츠만 변경하면 다른 화면도 손쉽게 구성 가능
- 프로젝트 구조 표준화로 협업 시 혼선 방지



CSS, Java Script 파일 저장하는 곳

여기도 **HTML** 파일 구조처럼 분리할 예정.

현재 디자인 작업이 들어가지 않으므로 미기재.



개발하면서 기재할 사항 있으면 추후 작성 예정

현재는 작성할 내용 없으므로 미기재

2025.05 - 2주차 현황

5. 프론트, 백엔드의 구조적 분리 및 GitHub 관리

✅ 5.1 프로젝트 구조 및 개발도구

- 전체 프로젝트는 **Spring Boot** 기반 백엔드로 구성되어 있으며
  정적 자원은 **src/main/resources/templates** 하위에 배치됩니다.

CSS, Java Script : templates/static/ HTML: templates/

- 프론트엔드의 독립성과 효율적인 개발을 위해 **Cursor IDE**를 사용하여
  프론트엔드 **HTML, CSS, JS** 등 **UI** 템플릿을 별도로 개발하게 됩니다
- **GitHub**를 통한 형상관리를 통해 협업 및 변경 이력 추적을 용이하게 하였습니다

✅ 5.2 프론트엔드와 백엔드의 분리 전략 I

- 프론트엔드를 백엔드(**Spring Boot**) 프로젝트 내부에서 직접 다루지 않고,
  독립적인 폴더(**capstone-templates**)로 구성하여 관리하였습니다.
- **Spring Boot**에서 사용하는 **templates** 디렉토리 구조와 동일하게 구성되어 있어,
  추후 통합 시 자연스럽게 연결될 수 있도록 설계하였습니다.

✅ 5.2 프론트엔드와 백엔드의 분리 전략 II

- 백엔드: **Spring Boot (IntelliJ** 사용)
- 프론트엔드: **HTML** 템플릿 중심 개발을 위해 **Cursor** 사용 (경량, 직관적, 빠른 개발에 유리)

무거운 백엔드 환경 없이 **HTML, CSS, JS** 작업만 빠르게 수행 가능하여 속도와 생산성 확보에 유리함.

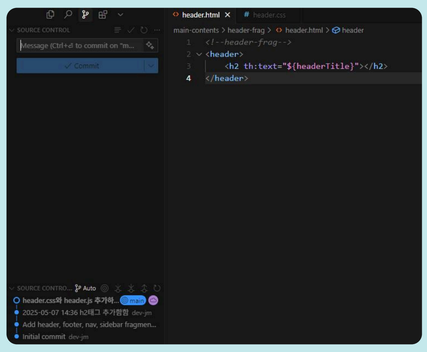
- 백엔드 프로젝트는 **Spring**의 **templates** 구조를 따름
- 프론트엔드는 같은 구조를 가진 별도 폴더(**capstone-templates**)로 관리
- 프론트 전용 **GitHub** 저장소 생성 → 추후 백엔드 통합 시 유연하게 병합 가능

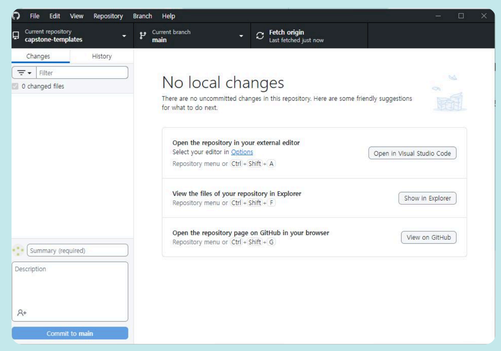
CURSOR IntelliJ IDEA

- 병렬 작업 가능 → 협업 효율 증가
- 문제 발생 시 영향 최소화

✅ 5.3 GitHub 저장소 구성

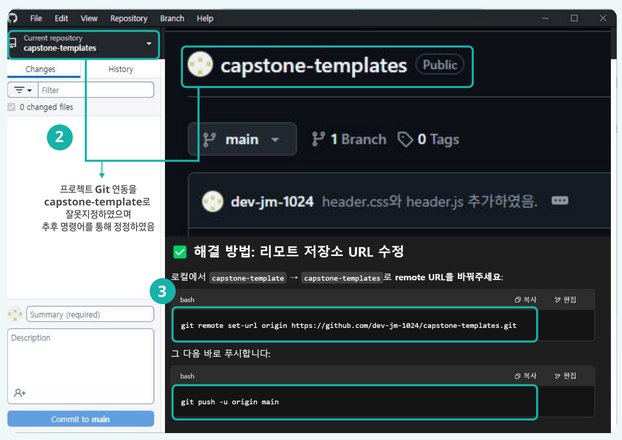
- 백엔드 전체 프로젝트는 별도의 **GitHub** 저장소에서 관리 중이며,
  BACK-END 개발과 **API** 중심의 로직을 포함합니다.
- 프론트엔드는 **capstone-templates**라는 이름의 독립 **GitHub** 저장소에 업로드되어
  별도 관리되며, 실제 배포 시 **templates** 폴더에 병합될 예정입니다.
- 로컬에서는 **Git remote URL**을 설정하고, **VS Code** 및 **GitHub Desktop**을 통해
  버전 관리를 수행하였습니다.

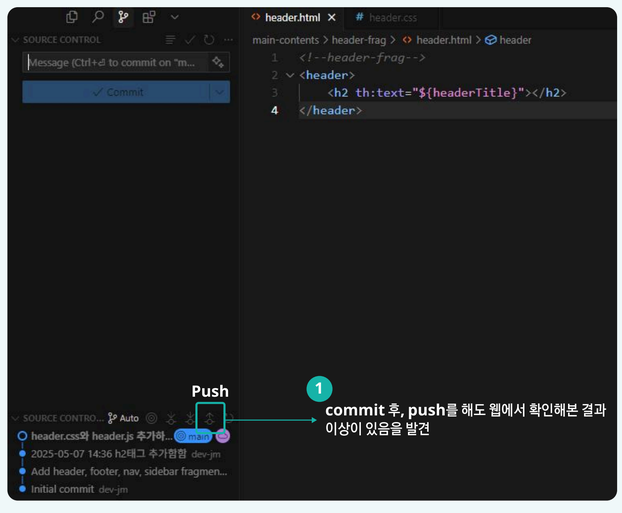




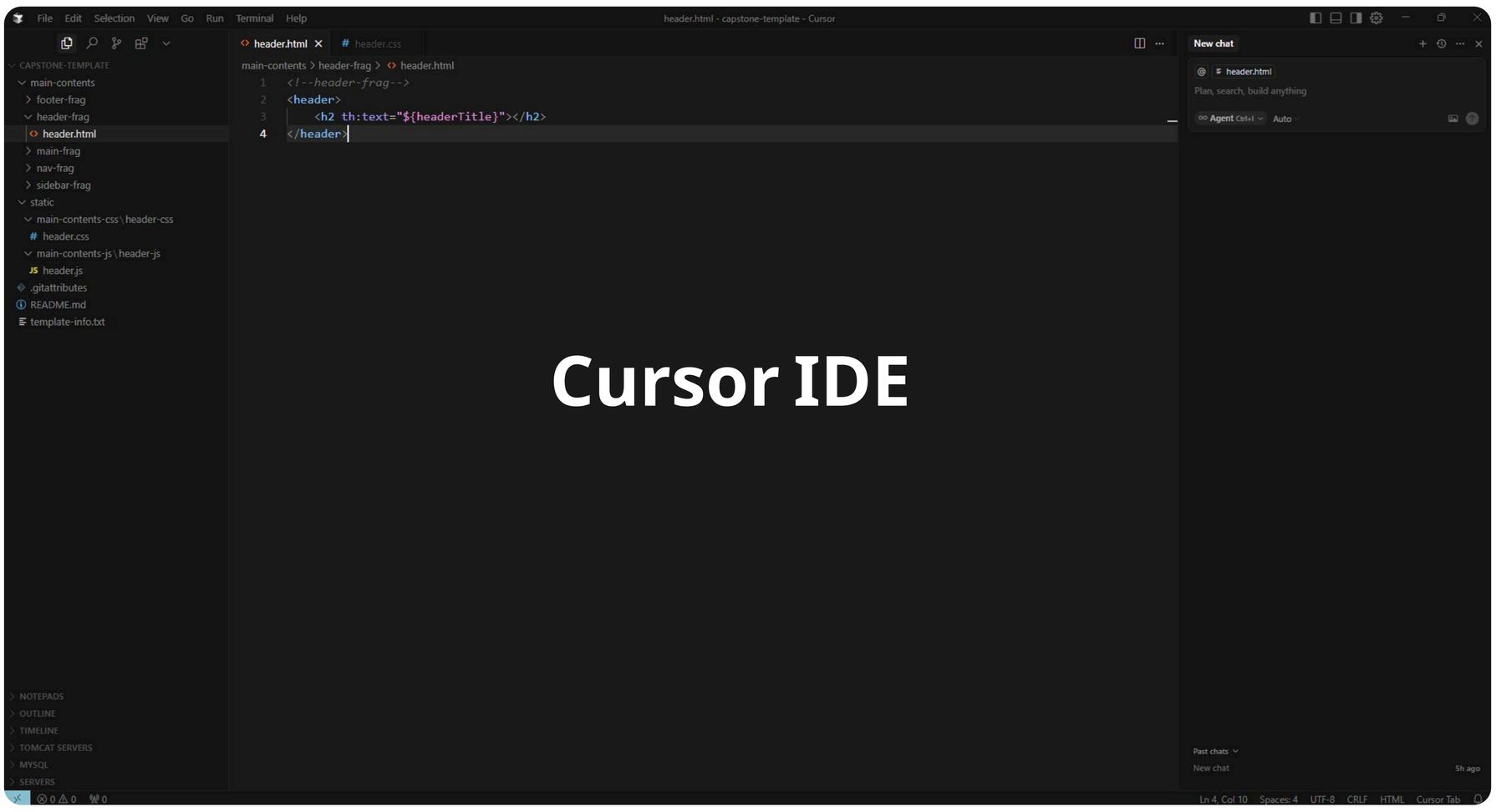
✅ 5.4 개발 및 관리 흐름

1. **capstone-templates** 폴더 생성
2. **Spring Boot**의 **templates** 구조를
   그대로 복제하여 구조 통일성 유지
3. **Cursor** 를 통해 **HTML** 파일 작업 수행
4. **Git**을 사용해 로컬 커밋 및 원격 저장소 **Push**
5. **GitHub** 저장소 주소 설정 실수로 인해 저장소 수정
   : capstone-templates.git로 **remote set-url** 명령어로 수정
6. 최종적으로 **GitHub**에 반영 완료 및 구조 분리 완료

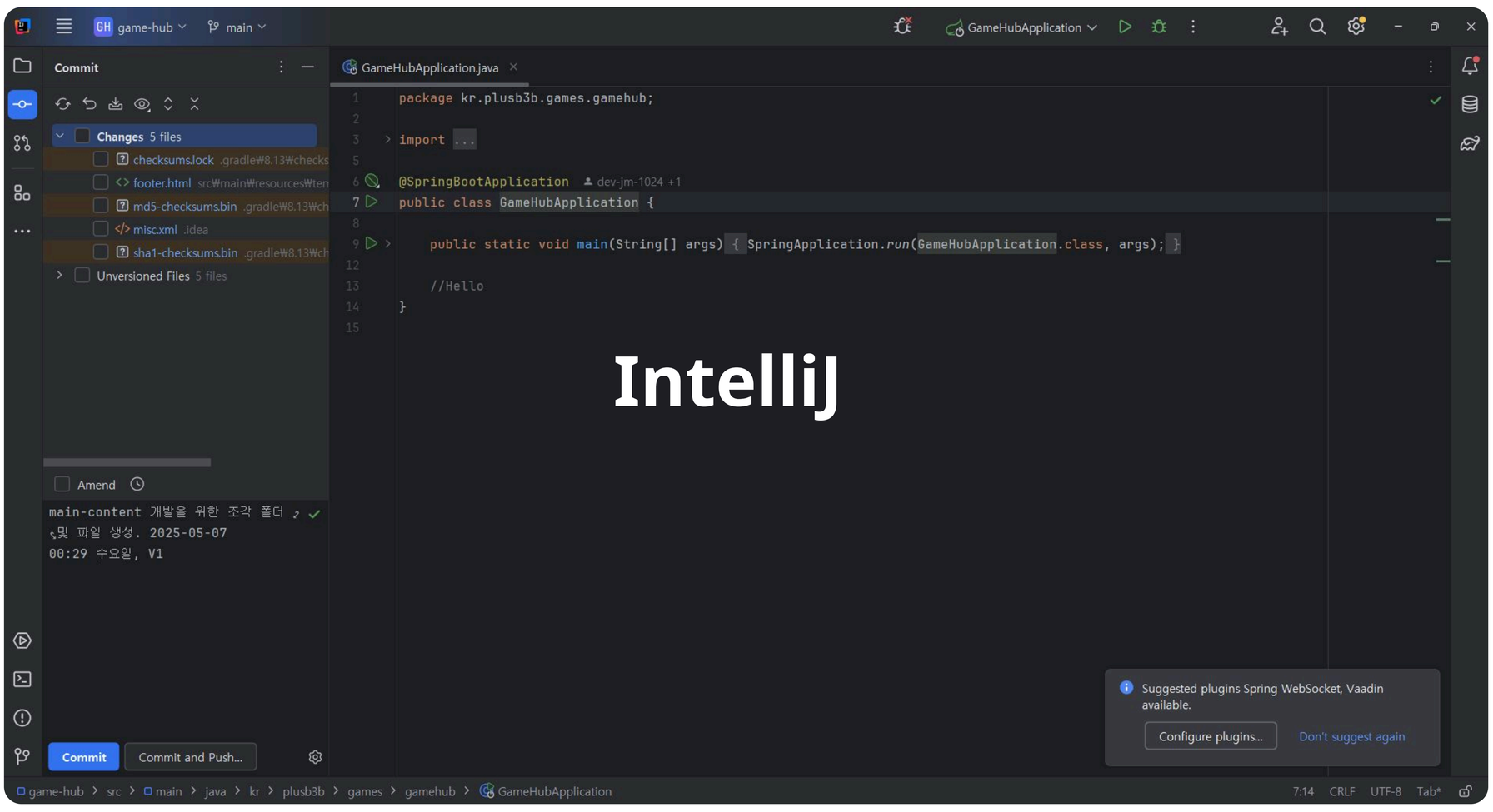




2025.05 - 2주차 현황



Cursor IDE



IntelliJ

6. 깃허브 링크

- **FRONT - END**
  : https://github.com/dev-jm-1024/capstone-templates.git
- **BACK - END**
  : https://github.com/dev-jm-1024/game-hub.git

2025.05 - 2주차 현황

7. 프론트엔드 UI/UX 설계 방향조사

✅ 7.1 모바일 및 PC 환경 대응

- 게임은 웹 기반으로 실행되기 때문에 **PC** 환경 중심의 레이아웃으로 구성
- 하지만 다음 기능들은 모바일에서도 원활히 사용될 수 있도록 반응형으로 설계함

게시판, 로그인/로그아웃, 프로필 커스터마이징, 게임 참고 자료 열람

- 기기 유형(**PC**/모바일)에 따라 접근을 제어하여, 모바일에서 게임 실행 페이지에 접근 시:

사용자에게 경고 메시지를 제공 | 이전 페이지로 자동 리디렉션 처리

- 모바일 '**PC**로 보기' 기능도 함께 지원하여 사용자 선택권 보장

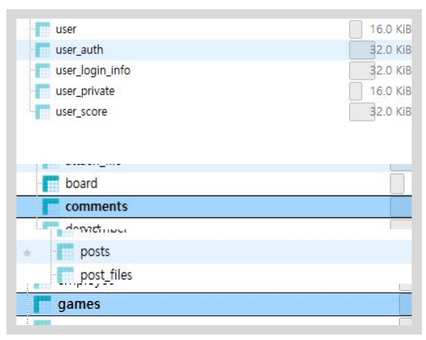
예시소스

```
<html>
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta name="color-scheme" content="light dark">
<script type="text/javascript">if (!window.T) { window.T = {} }
if (/Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera
Mini/i.test(navigator.userAgent)) { // Mobile
  console.log('true')
} else { // PC
  console.log('false')
}

function isMobile(){
    const userAgent = navigator.userAgent;
    if (userAgent.match(/iPhone|iPod|Android|Windows
CE|BlackBerry|Symbian|Windows Phone|webOS|Opera Mini|Opera
Mobi|POLARIS|IEMobile|lgtelecom|nokia|SonyEricsson/i) != null ||
userAgent.match(/LG|SAMSUNG|Samsung/) != null) {
        return true;
    } else {
        return false;
    }
}

if(isMobile()) {
alert('isMobile()') // Mobile
console.log('isMobile()')
   // mobile only code

} else { // PC
alert('isPc()')
console.log('isPc()')
   // pc only code
}</script>
</head>
<body><div class="line-gutter-backdrop"></div>
</body>

</html>
```

8. 백엔드 개발 및 DB 설계 - 구현 ver

✅ 8.1 SQL문 생성과 DB 정보 등록

- 기존 데이터베이스 설계를 검토 및 수정한 후, 이를 기반으로 **SQL** 문으로 테이블 생성
- 프로젝트의 **application.yml** 파일에 **DB** 연결 정보 등록하여 연동 준비 완료



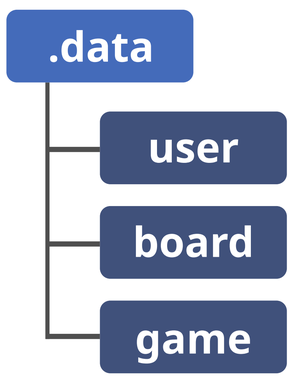

```
sql
CREATE TABLE games (
    game_id BIGINT PRIMARY KEY COMMENT '게임 고유 ID',
    board_id BIGINT COMMENT '게시판 ID',
    mb_id BIGINT COMMENT '업로더 사용자 ID',
    game_name VARCHAR(200) COMMENT '게임 이름',
    game_description TEXT COMMENT '게임 소개',
    team_name VARCHAR(100) COMMENT '팀/기자 또는 팀 이름',
    specs TEXT COMMENT '권장 사양 (CPU, RAM 등)',
    game_file_url VARCHAR(500) COMMENT '게임 파일 경로 또는 URL',
    created_at DATE COMMENT '게임 등록일',
    is_visible INT DEFAULT 1 COMMENT '노출 여부 (1: 노출, 0: 숨김)',
    game_version VARCHAR(50) COMMENT '게임 버전 (예: 1.0.0)',
    genre VARCHAR(100) COMMENT '게임 장르 (예: RPG, 퍼즐 등)',
    platform VARCHAR(100) COMMENT '지원 플랫폼 (PC, Mobile 등)',
    game_hash VARCHAR(255) COMMENT '게임 파일 무결성 확인용 해시값 (SHA256 등)',
    is_approved INT DEFAULT 0 COMMENT '운영자 승인 여부 (1: 승인, 0: 미승인)',
    extra1 VARCHAR(255), extra2 VARCHAR(255), extra3 VARCHAR(255), extra4 VARCHAR(255), extra5 VA
    extra6 VARCHAR(255), extra7 VARCHAR(255), extra8 VARCHAR(255), extra9 VARCHAR(255), extra10 V
    FOREIGN KEY (board_id) REFERENCES board(board_id),
    FOREIGN KEY (mb_id) REFERENCES user(mb_id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

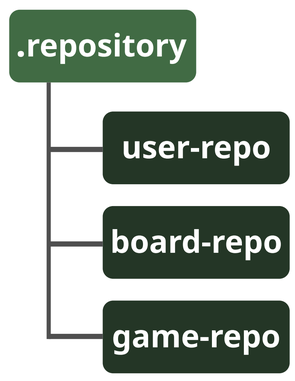
```
# ========= Database Connection =========
spring.datasource.url=jdbc:mariadb://<HOST>:<PORT>/<DB_NAME>?useUnicode=true&characterEncoding=utf8mb4&serverTimezone=Asia/Seoul
spring.datasource.username=<DB_USERNAME>
spring.datasource.password=<DB_PASSWORD>
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver

# ========= JPA / Hibernate =========
spring.jpa.hibernate.ddl-auto=none


# or validate / update / create / create-drop
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MariaDB103Dialect
spring.jpa.properties.hibernate.format_sql=true

# ========= Connection Pool (Optional - HikariCP) =========
spring.datasource.hikari.maximum-pool-size=10
spring.datasource.hikari.connection-timeout=20000
spring.datasource.hikari.minimum-idle=5
spring.datasource.hikari.idle-timeout=300000
spring.datasource.hikari.max-lifetime=1800000

# ========= Logging (Optional) =========
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

9. 프로젝트 파일 및 패키지 구조

✅ 9.1 파일구조 설계

- 전체 **Java** 코드 구조는 **data** 패키지를 루트로 두고 기능별로 하위 패키지를 구성
- **JPA** 연동을 위해 **repository** 패키지를 별도 생성하고, 기능별로 인터페이스를 나눌 계획





10. 도메인 클래스 구성 방식

✅ 10.1 도메인 클래스 작성

- 테이블을 기준으로 도메인 클래스를 작성하였으며, 각 클래스는 다음의 구성 요소를 포함

필드 정의 | getter/setter method | 매개변수 없는 기본 생성자

- 추후 **JPA** 연동을 위한 엔티티로 확장 예정 (**@Entity, @Id** 등 추가)

Board.java | PostFiles.java | Posts.java | Comments.java

```
package kr.plusb3b.games.gamehub.data.board;

import java.time.LocalDateTime;

public class PostFiles {  no usages  dev-jm

    //파일 고유 아이디
    private Long file_id;  2 usages

    //파일이 속한 게시글 아이디 : 외래키
    private Long post_id;  2 usages

    //파일경로 또는 이름
    private String file_url;  2 usages

    //파일 유형
    private String file_type;  2 usages

    //업로드 날짜
    private LocalDateTime upload_date;  2 usages

    //여분컬럼1~10
    private String file_extra1;  2 usages
    private String file_extra2;  2 usages
    private String file_extra3;  2 usages
    private String file_extra4;  2 usages
    private String file_extra5;  2 usages
    private String file_extra6;  2 usages
    private String file_extra7;  2 usages
    private String file_extra8;  2 usages
    private String file_extra9;  2 usages
    private String file_extra10;  2 usages
```