System design document
Mlb API

Pourpose

This is system design document describe the system and architecture for mlb api. Mlb api is designed for anyone who is online can use mlb players information from last year.

System overview

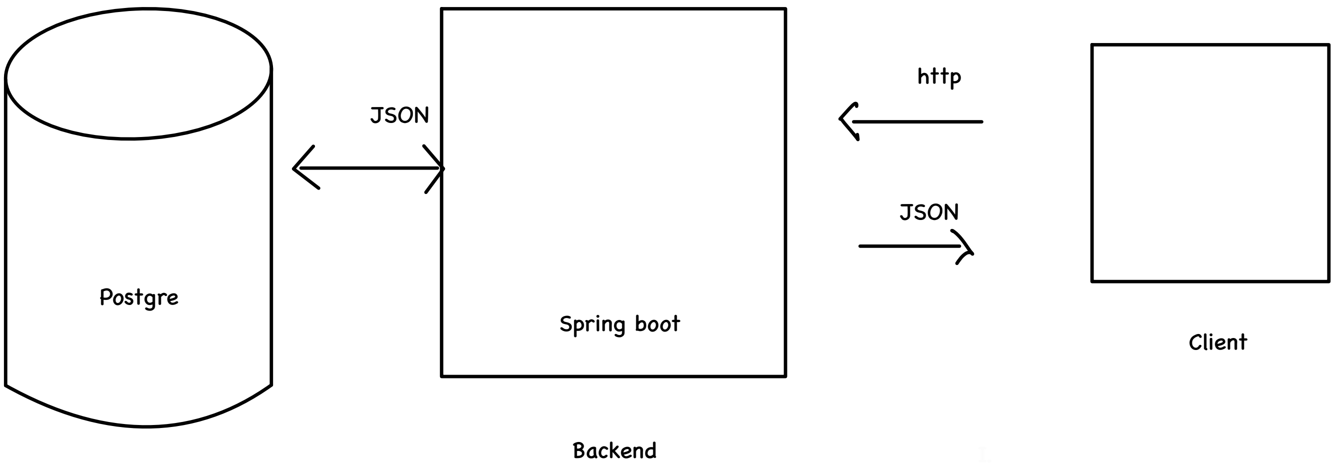


Figure I

Figure I above represents the architectural structure I have chosen for mlb api. The backend will handle HTTP get request and pulling right data from postgresql database then return to the client.

System components

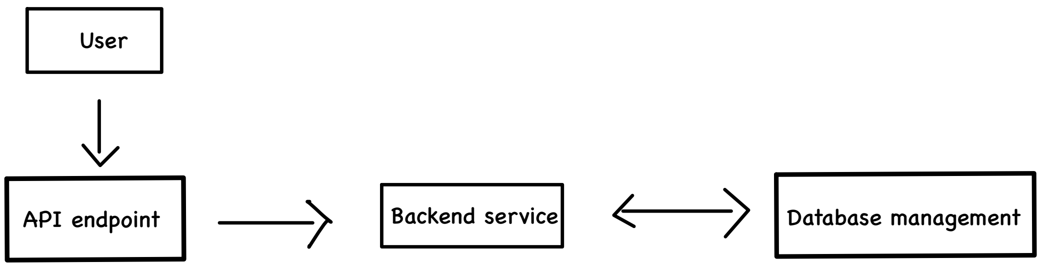
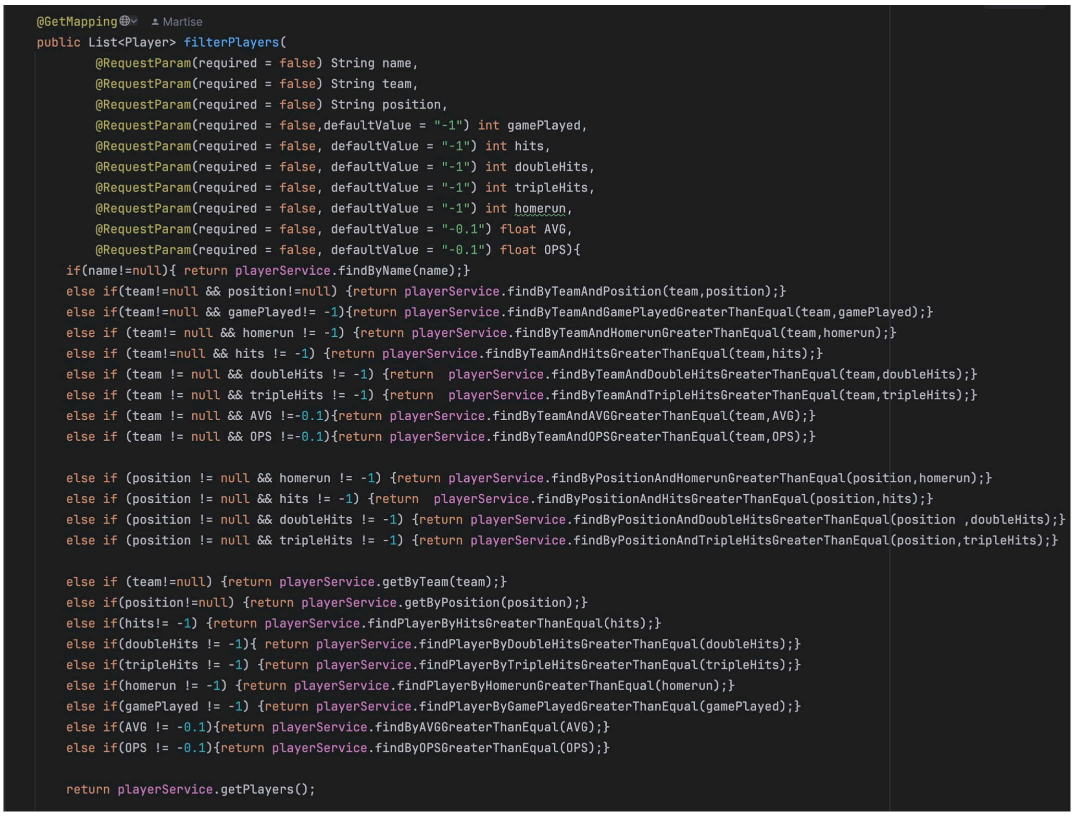


Figure2 above showing how my api expected to work and how components interact.

Workflow

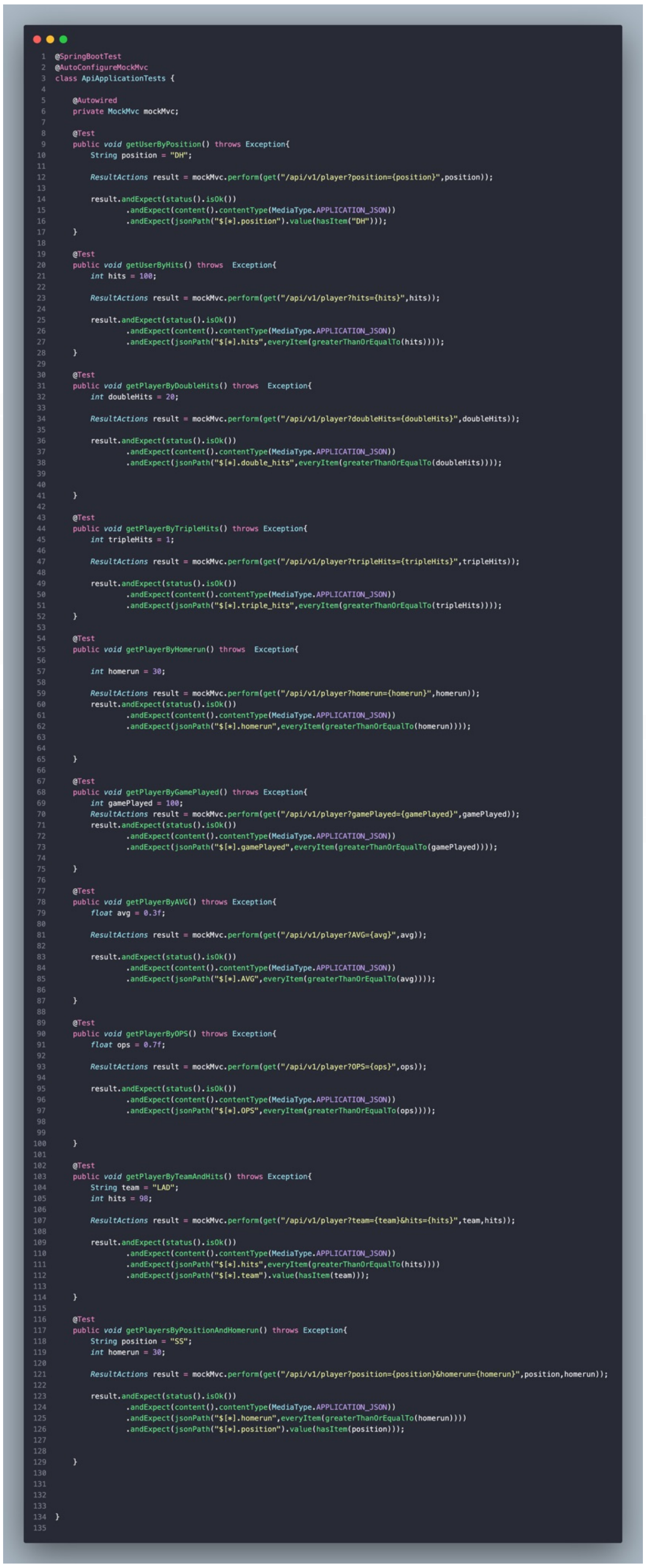
User sent HTTP get request then playerControll.java class handles it then it access to playerService class and playerRepository class it returns data.
The reason why I didn't give path to @GetMapping("/team/{team}") like this is I have various variable in Player class and many permutation so I tried to keep it short as I can.



```java
@GetMapping   Martise
public List<Player> filterPlayers(
        @RequestParam(required = false) String name,
        @RequestParam(required = false) String team,
        @RequestParam(required = false) String position,
        @RequestParam(required = false,defaultValue = "-1") int gamePlayed,
        @RequestParam(required = false, defaultValue = "-1") int hits,
        @RequestParam(required = false, defaultValue = "-1") int doubleHits,
        @RequestParam(required = false, defaultValue = "-1") int tripleHits,
        @RequestParam(required = false, defaultValue = "-1") int homerun,
        @RequestParam(required = false, defaultValue = "-0.1") float AVG,
        @RequestParam(required = false, defaultValue = "-0.1") float OPS){
    if(name!=null){ return playerService.findByName(name);}
    else if(team!=null && position!=null) {return playerService.findByTeamAndPosition(team,position);}
    else if(team!=null && gamePlayed!= -1){return playerService.findByTeamAndGamePlayedGreaterThanEqual(team,gamePlayed);}
    else if (team!= null && homerun != -1) {return playerService.findByTeamAndHomerunGreaterThanEqual(team,homerun);}
    else if (team!=null && hits != -1) {return playerService.findByTeamAndHitsGreaterThanEqual(team,hits);}
    else if (team != null && doubleHits != -1) {return  playerService.findByTeamAndDoubleHitsGreaterThanEqual(team,doubleHits);}
    else if (team != null && tripleHits != -1) {return  playerService.findByTeamAndTripleHitsGreaterThanEqual(team,tripleHits);}
    else if (team != null && AVG !=-0.1){return playerService.findByTeamAndAVGGreaterThanEqual(team,AVG);}
    else if (team != null && OPS !=-0.1){return playerService.findByTeamAndOPSGreaterThanEqual(team,OPS);}

    else if (position != null && homerun != -1) {return playerService.findByPositionAndHomerunGreaterThanEqual(position,homerun);}
    else if (position != null && hits != -1) {return  playerService.findByPositionAndHitsGreaterThanEqual(position,hits);}
    else if (position != null && doubleHits != -1) {return playerService.findByPositionAndDoubleHitsGreaterThanEqual(position ,doubleHits);}
    else if (position != null && tripleHits != -1) {return playerService.findByPositionAndTripleHitsGreaterThanEqual(position,tripleHits);}

    else if (team!=null) {return playerService.getByTeam(team);}
    else if(position!=null) {return playerService.getByPosition(position);}
    else if(hits!= -1) {return playerService.findPlayerByHitsGreaterThanEqual(hits);}
    else if(doubleHits != -1){ return playerService.findPlayerByDoubleHitsGreaterThanEqual(doubleHits);}
    else if(tripleHits != -1) {return playerService.findPlayerByTripleHitsGreaterThanEqual(tripleHits);}
    else if(homerun != -1) {return playerService.findPlayerByHomerunGreaterThanEqual(homerun);}
    else if(gamePlayed != -1) {return playerService.findPlayerByGamePlayedGreaterThanEqual(gamePlayed);}
    else if(AVG != -0.1){return playerService.findByAVGGreaterThanEqual(AVG);}
    else if(OPS != -0.1){return playerService.findByOPSGreaterThanEqual(OPS);}

    return playerService.getPlayers();
```

Test

I use spring boot test To test my API returns data, I test each filter work then test each combination



```java
@SpringBootTest
@AutoConfigureMockMvc
class ApiApplicationTests {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void getUserByPosition() throws Exception{
        String position = "DH";

        ResultActions result = mockMvc.perform(get("/api/v1/player?position={position}",position));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].position").value(hasItem("DH")));
    }

    @Test
    public void getUserByHits() throws  Exception{
        int hits = 100;

        ResultActions result = mockMvc.perform(get("/api/v1/player?hits={hits}",hits));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].hits",everyItem(greaterThanOrEqualTo(hits))));
    }

    @Test
    public void getPlayerByDoubleHits() throws  Exception{
        int doubleHits = 20;

        ResultActions result = mockMvc.perform(get("/api/v1/player?doubleHits={doubleHits}",doubleHits));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].double_hits",everyItem(greaterThanOrEqualTo(doubleHits))));


    }

    @Test
    public void getPlayerByTripleHits() throws Exception{
        int tripleHits = 1;

        ResultActions result = mockMvc.perform(get("/api/v1/player?tripleHits={tripleHits}",tripleHits));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].triple_hits",everyItem(greaterThanOrEqualTo(tripleHits))));
    }

    @Test
    public void getPlayerByHomerun() throws  Exception{

        int homerun = 30;

        ResultActions result = mockMvc.perform(get("/api/v1/player?homerun={homerun}",homerun));
        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].homerun",everyItem(greaterThanOrEqualTo(homerun))));


    }

    @Test
    public void getPlayerByGamePlayed() throws Exception{
        int gamePlayed = 100;
        ResultActions result = mockMvc.perform(get("/api/v1/player?gamePlayed={gamePlayed}",gamePlayed));
        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].gamePlayed",everyItem(greaterThanOrEqualTo(gamePlayed))));

    }

    @Test
    public void getPlayerByAVG() throws Exception{
        float avg = 0.3f;

        ResultActions result = mockMvc.perform(get("/api/v1/player?AVG={avg}",avg));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].AVG",everyItem(greaterThanOrEqualTo(avg))));

    }

    @Test
    public void getPlayerByOPS() throws Exception{
        float ops = 0.7f;

        ResultActions result = mockMvc.perform(get("/api/v1/player?OPS={ops}",ops));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].OPS",everyItem(greaterThanOrEqualTo(ops))));


    }

    @Test
    public void getPlayerByTeamAndHits() throws Exception{
        String team = "LAD";
        int hits = 98;

        ResultActions result = mockMvc.perform(get("/api/v1/player?team={team}&hits={hits}",team,hits));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].hits",everyItem(greaterThanOrEqualTo(hits))))
                .andExpect(jsonPath("$[*].team").value(hasItem(team)));

    }

    @Test
    public void getPlayersByPositionAndHomerun() throws Exception{
        String position = "SS";
        int homerun = 30;

        ResultActions result = mockMvc.perform(get("/api/v1/player?position={position}&homerun={homerun}",position,homerun));

        result.andExpect(status().isOk())
                .andExpect(content().contentType(MediaType.APPLICATION_JSON))
                .andExpect(jsonPath("$[*].homerun",everyItem(greaterThanOrEqualTo(homerun))))
                .andExpect(jsonPath("$[*].position").value(hasItem(position)));


    }




}
```