1 Introdução

Serve o presente documento para elucidar o conjunto de alterações efetuadas a nível da solução, ou seja, alterações de código, que o grupo considerou relevantes. Constam também as respetivas justificações e de que forma essas alterações impactaram, ou que vieram concretizar.

2 Alterações efetuadas

2.1 Adição de condição de inserção de prognóstico no boletim

Neste sentido, foi adicionada esta condição de inserção de um prognóstico no boletim: limite máximo de 20 prognósticos, ou seja, não há limite de apostas que podem ser feitas, mas o conjunto de prognósticos só podem ser até 20. Para isto, para se adicionaram duas linhas de informação, que se seguem:

```
[...]
    else if(boletim.size() == 20)
                        this.view.line("Nao pode apostar
                        em mais de 20 jogos!");
[...]
```

2.2 Adição da tabela *Observer*

Neste sentido, foi adicionada a tabela *Observer*, que contém o *id* do jogo e o *id* do *user* que está interessado no respeito jogo. Esta tabela é fundamental para a inclusão do *design pattern*. Não houve necessidade de criação de uma tabela para os observados, uma vez que essa relação já é evidenciada nesta mesma tabela, que associa os *ids* do jogo ao utilizador. Seguem as principais linhas adicionadas/alteradas:

```
public void run() {
        int opcao = -1;
        while (opcao != 0) {
            this.view.apostadorMainMenu(this.user.getNomeutilizador() +
            " - " + this.user.getSaldo()
            + this.currencyTime.getCurrency()); //mudar de acordo com regiao
            opcao = this.scanOption(0, 10);
            switch (opcao) {
                case 1 -> {
                    //Observa os jogos
                    this.view.subheader("Jogos", this.user.getNomeutilizador() +
                    " - " + this.user.getSaldo()
                    + this.currencyTime.getCurrency());
                    String desporto = this.getDesporto();
                    if (desporto != null) {
                        List<Jogo> jogos = this.db.getJogos(desporto);
                        for (int i = 0; i < jogos.size(); i++) {
                            this.view.line((i + 1) + " - " + jogos.get(i).toString() + "\n");
                        }

                        this.view.optionsMenu(new String[]
                        {"1 - Seguir Jogo", "2 - Deixar de Seguir Jogo", "0 - Voltar"});
                        int opcao2 = this.scanOption(0, 2);
                        if (opcao2 == 1) {
                            this.view.line("Qual o jogo que deseja seguir?
                            (0 para voltar)");
                            int jogo = this.scanOption(0, jogos.size());
                            if (jogo!= 0) {
                                this.db.seguirJogo(this.user.getNomeutilizador(), jogos.get(jogo - 1).getId());
                                this.view.line("Jogo seguido com sucesso!");
                            }
                        }
                        else if (opcao2 == 2) {
                            this.view.line("Qual o jogo que deseja deixar de seguir? (0 para voltar)");
                            jogos = this.db.getJogosSeguidos(this.user.getNomeutilizador());
                            int jogo = this.scanOption(0, jogos.size());
                            if (jogo!= 0) {
                                this.db.deixarSeguirJogo(this.user.getNomeutilizador(), jogos.get(jogo - 1).getId());
                                this.view.line("Jogo deixado de seguir com sucesso!");
                            }
                        }
                    }
                }
[...]
```

3 Conclusão

Nesta terceira fase do projeto, seguimos de igual forma as técnicas de análise de requisitos e os padrões de arquitetura e desenvolvimento neste que foi um dos nossos primeiros contactos com a Engenharia de *Software*, no que diz respeito a uma análise muito mais profunda dos requisitos de *Software*.

Consideramos que este tema permitiu expandir os nossos horizontes para os *Design Patterns*, boa prática fundamental uma vez que fornecem soluções para problemas comuns de design de software que surgem ao longo do desenvolvimento de um projeto. Além disso, podem ajudar a manter o código limpo e fácil de manter, e aumentar a reutilização de código.