Relatório de Cobertura de Testes

Era necessário ter uma cobertura superior a 50% de testes JUnit em pelo menos uma métrica. A métrica escolhida pelo nosso grupo foi a de *Line Counters (Covered Lines)*.

Com a utilização do método setUp(), conseguíamos definir as variáveis necessárias aos testes. Através do plug-in EclEmma, conseguíamos averiguar a percentagem da cobertura dos testes. A realização destes testes teve com base o método AssertEquals().

Ao ter em conta que este projeto tinha uma grande parte gráfica (Front-End), todos os packages que estão presentes apenas para a criação dos calendários, não foram submetidos aos testes JUnit (com algumas exceções que serão mais à frente explicadas). Isto acontece pois estes packages foram não criadas por nós e apenas adicionámos algumas classes para a leitura dos ficheiros . Os packages que fazem parte do Front-End do projeto, não são essenciais ao projeto, visto que os horários dos membros e da reunião são devolvidos apenas através das classes no package de Back-End. No entanto, há classes dentro de Front-End Packages que são necessárias para a leitura do ficheiro de Back-End para o Front-End, daí termos considerado importantes realizar testes sobre elas também.

Front-End Packages:

- Calendar (Exceção das classes que são de Back-End: CalendarEmptyClickEvent, CalendarEventClickEvent, CalendarEvent e Calendar).
- **DayCalendar** (Exceção da classe que é de Back-End: **DayCalendar**).
- MonthCalendar (Exceção da classe que é de Back-End: ToDayTest).
- WeekCalendar (Exceção das classes que são de Back-End: Week e WeekCalendar).
- Menu

Back-End Package:

• ES_2022_LETI_Grupo_15.Projeto_ES

E Coverage ×									
Element	Coverage	Covered Lines		Total Lines					
✓	21.1 %	430	1,607	2,037					
> 🔑 src/main/java	16.1 %	303	1,574	1,877					
> 👺 src/test/java	79.4 %	127	33	160					

Figura 1.

Percentagem cobertura de testes (a percentagem muda devido ao facto de apenas o Back-End ter sido realizado).

Element	Coverage	Covered Lines	Missed Lines	Total Lines
✓	21.1 %	430	1,607	2,037
# src/main/java	16.1 %	303	1,574	1,877
> # MonthCalendar	1.4 %	9	652	661
> # Menu	0.0 %	0	555	555
> 👪 Calendar	16.6 %	51	257	308
> # WeekCalendar	32.5 %	25	52	77
> # DayCalendar	19.0 %	8	34	42
> # ES_2022_LETI_Grupo_15.P	89.7 %	210	24	234
> 👺 src/test/java	79.4 %	127	33	160

Figura 2.

Percentagem cobertura de testes por packages.

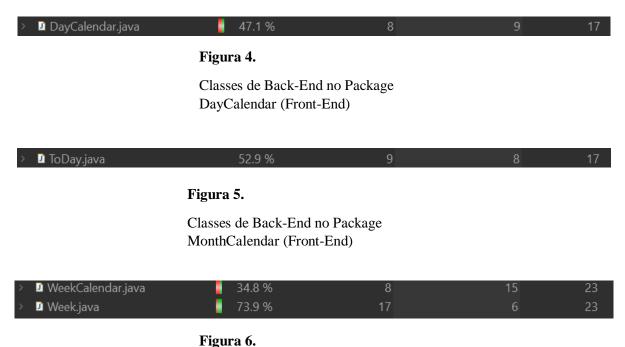
Observando a **Figura 2.**, conseguimos observar que a percentagem de testes do package de Back-End é bastante mais elevada.

Assim, vamos juntar o número de linhas das classes dentro dos packages de Front-End que considerámos importantes realizar os testes e iremos verificar a percentagem.

>	Calendar Empty Click Event. java	100.0 %	4	0	4
>	Calendar Event Click Event. java	100.0 %	4	0	4
>	Calendar.java	6.7 %	15	209	224
>	CalendarEvent.java	53.2 %	25	22	47

Figura 3.

Classes de Back-End no Package Calendar (Front-End)



Classes de Back-End no Package WeekCalendar (Front-End)

Após observação das **Figuras 3. a 6.**, conseguimos observar os seguintes dados relativos ao Package e classes de Back-End (que foram fruto de testes JUnit):

Total Lines: 593
Coverage Lines: 300
Missed Lines: 293
Figura 7.
Dados a serem usados para cálculo de percentagem (Back-End)

E após observação da **Figura 1.,** podemos observar os seguintes dados relativos aos Packages de Front-End (com exceção das classes Back-End neles inseridas).

Total Lines: 1284
Coverage Lines: 0
Missed Lines: 1284
Figura 8.
Dados a não serem usados para cálculo de percentagem (Front-End)

Assim, para a "src/main/java", a **coverage** (%), não será os **16.1%**, mas sim (300/593) * 100 = **50.6** %, estando acima dos **50** % pedidos.