NOVA SCHOOL OF SCIENCE & TECHNOLOGY

THE S TEAM

Trabalho Realizado por:

Carlos Soares - 57009
Pedro Inácio - 59184
Martin Magdalinchev - 58172
Bernardo Atalaia - 59962
Francisco Parrinha - 58360

Link do YouTube: https://youtu.be/g9VEqnSQbKk
GitHub link: https://github.com/f-parrinha/ganttproject-ES

Software Engineering - GanttProject

Índice

Introdução

O trabalho apresentado pela cadeira de Engenharia de Software está dividido em duas fases e assenta sobre o projeto *open source*, já em distribuição, GanttProject. O relatório é sobre a segunda fase. Neste estágio, é proposto que uma equipa de cinco elementos implemente por cima da aplicação duas novas funcionalidades.

Para as duas novas implementações, foram sugeridas as seguinte ideias: adicionar dois novos painéis; um para um *burndown chart* (chamado *Burndown Chart*) e um para um *Planner*, que apresenta algumas estatísticas sobre o projeto montado na aplicação. As duas foram implementadas com êxito e na sua totalidade.

A integração das funcionalidades foi realizada de forma a manter a extensibilidade de todo o código já presente para a aplicação *GanttProject*, sem ter existido necessidade de fazer alterações, com exceção da correção de um bug existente na estrutura do próprio projeto. Este erro será discutido com maior pormenor mais à frente no relatório.

É importante notar que a estrutura pedida no enunciado encontra-se finalizada na *branch* BRANCH_2_8_9. Algumas pré configurações são necessárias para que o build do projeto corra com sucesso. Para correr pela primeira vez é necessário realizar os seguintes passos no terminal, depois do clone, já dentro da pasta do projeto: git checkout BRANCH_2_8_9; git remote add <name> <link do repositório dos Professores>; git fetch <name>. Agora já é possível realizar o build, sem erros, na pasta *ganttproject-builder*.

Preâmbulo

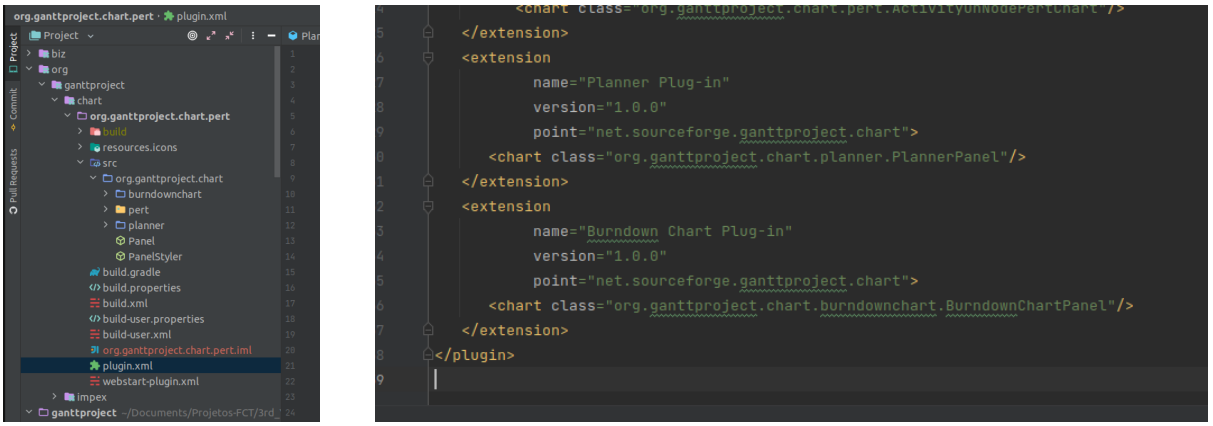
Como dito anteriormente, as duas novas ferramentas encontram-se à parte da estrutura principal da aplicação, garantindo a sua extensibilidade e modularidade. Para que tal fosse possível, foi essencial uma pesquisa mais aprofundada sobre a organização do código original. Neste estudo, foi descoberto que o projeto já apresentava uma ferramenta de *Plug-ins*, à qual a janela (ou *view*) *PERT Chart*, já utilizava. Esta ferramenta funciona fazendo *scrap* de um ficheiro chamado *plugin.xml*, que contém todas as novas extensões. Para adicionar as novas *views* (*Planner* e *Burndown Chart*), bastou adicionar dois novos campos com as indicações da localização das duas novas classes que implementam as novas *views*.

Neste modo, para aceder às novas funcionalidades, basta clicar no menu *View*, na barra superior, onde vão aparecer, para além do *PERT Chart*, o *Planner* e o *Burndown Chart*. Ao clicar em cada um destes botões, aparecerá uma nova *view* no ecrã principal. Para remover basta realizar os mesmo passos da adição, clicando, dentro do menu *View*, no botão com o nome do painel já adicionado

Durante a implementação, foram encontrados alguns desafios, sendo um deles um erro que existia dentro da própria estrutura de como as *views* estavam implementadas previamente no *GanttProject*. Em resumo, quando se pretendia remover um painel já adicionado, a indexação dos que ainda estavam abertos não era recalculado, por isso, sempre que não se removia de ordem inversa à adicionada, o programa eventualmente levantava a exceção *NullPointerException*. Para corrigir foi suficiente adicionar um re-cálculo dos índices de cada *view* na classe *ViewManager*.

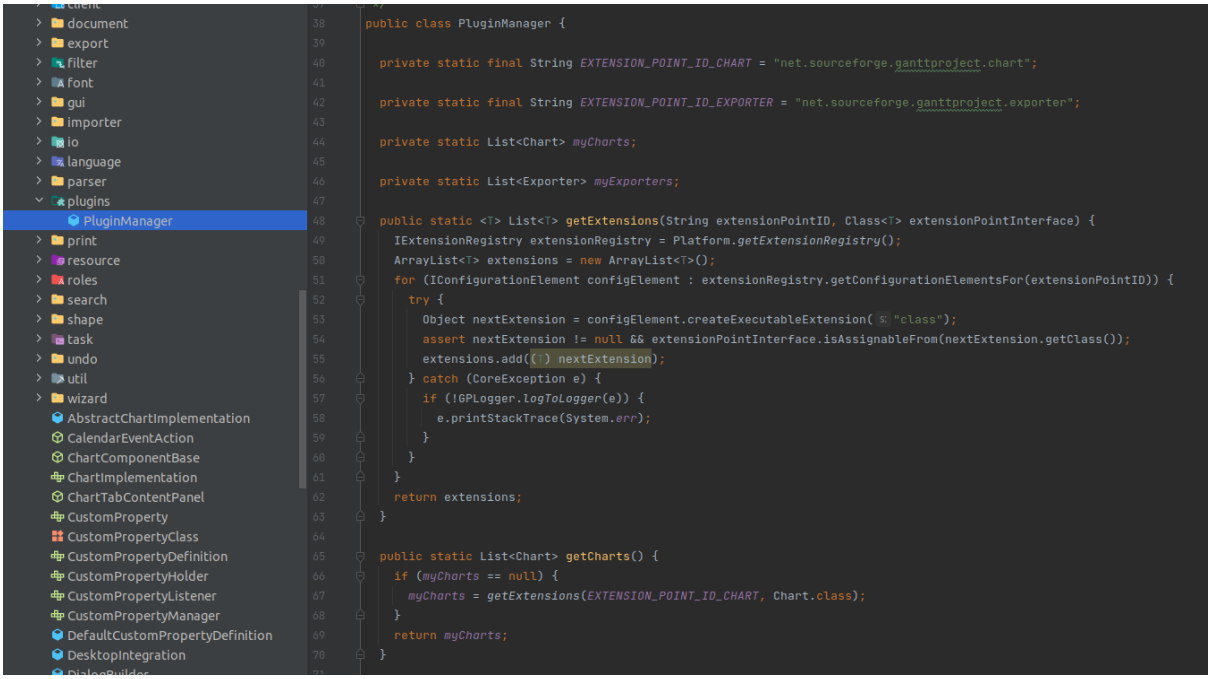
Em baixo seguem-se alguns *screenshots* a explicar, em código, o que foi dito anteriormente:

Figuras 1 - Código para a implementação à base de Plug-ins



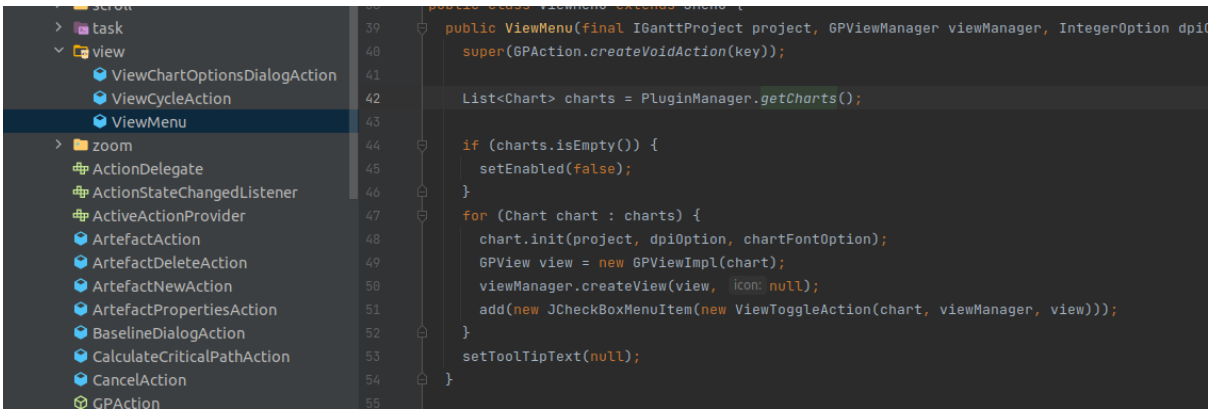
```
    </extension>
    <extension
            name="Planner Plug-in"
            version="1.0.0"
            point="net.sourceforge.ganttproject.chart">
       <chart class="org.ganttproject.chart.planner.PlannerPanel"/>
    </extension>
    <extension
            name="Burndown Chart Plug-in"
            version="1.0.0"
            point="net.sourceforge.ganttproject.chart">
       <chart class="org.ganttproject.chart.burndownchart.BurndownChartPanel"/>
    </extension>
</plugin>
```

Figuras 1.1: As novas extensões no ficheiro plugin.xml



```
public class PluginManager {

  private static final String EXTENSION_POINT_ID_CHART = "net.sourceforge.ganttproject.chart";

  private static final String EXTENSION_POINT_ID_EXPORTER = "net.sourceforge.ganttproject.exporter";

  private static List<Chart> myCharts;

  private static List<Exporter> myExporters;

  public static <T> List<T> getExtensions(String extensionPointID, Class<T> extensionPointInterface) {
    IExtensionRegistry extensionRegistry = Platform.getExtensionRegistry();
    ArrayList<T> extensions = new ArrayList<T>();
    for (IConfigurationElement configElement : extensionRegistry.getConfigurationElementsFor(extensionPointID)) {
      try {
        Object nextExtension = configElement.createExecutableExtension( s: "class");
        assert nextExtension != null && extensionPointInterface.isAssignableFrom(nextExtension.getClass());
        extensions.add( e: nextExtension);
      } catch (CoreException e) {
        if (!GPLogger.logToLogger(e)) {
          e.printStackTrace(System.err);
        }
      }
    }
    return extensions;
  }

  public static List<Chart> getCharts() {
    if (myCharts == null) {
      myCharts = getExtensions(EXTENSION_POINT_ID_CHART, Chart.class);
    }
    return myCharts;
  }
```

Figura 2.2: Plugin Manager. Onde carrega as extensões do ficheiro.xml. Já existia.



```
  public ViewMenu(final IGanttProject project, GPViewManager viewManager, IntegerOption dpiO
    super(GPAction.createVoidAction(key));

    List<Chart> charts = PluginManager.getCharts();

    if (charts.isEmpty()) {
      setEnabled(false);
    }
    for (Chart chart : charts) {
      chart.init(project, dpiOption, chartFontOption);
      GPView view = new GPViewImpl(chart);
      viewManager.createView(view,  icon: null);
      add(new JCheckBoxMenuItem(new ViewToggleAction(chart, viewManager, view)));
    }
    setToolTipText(null);
  }
```

Figura 2.3: Onde são carregadas as novas views. Já existia.

Figuras 2 - Código para a correção do problema encontrado

```
@Override
public void toggleVisible(GPView view) {
  ViewHolder viewHolder = myViews.get(view);
  assert viewHolder != null;

  /*
   * Stores the number of tabs because it will change in the next line
   *    if myTabs.getTabCount() is used directly in the
   *    'if' statement, the app will crash sometimes
   */
  int tabsCount = myTabs.getTabCount();
  viewHolder.setVisible(!viewHolder.isVisible());

  if (viewHolder.getMyIndex() != tabsCount) {
    decreaseIndexes(view);
  }
}
```

Figura 2.1: Código em ViewManagerImpl que recalculam os índices de cada view ainda presente

```
/**
 * Decreases the current index of the view
 */
public void decreaseIndex(){
    myIndex --;
}

/**
 * Gets the current value of the view's index
 *
 * @return index
 */
public int getMyIndex() {
  return myIndex;
}
}
```

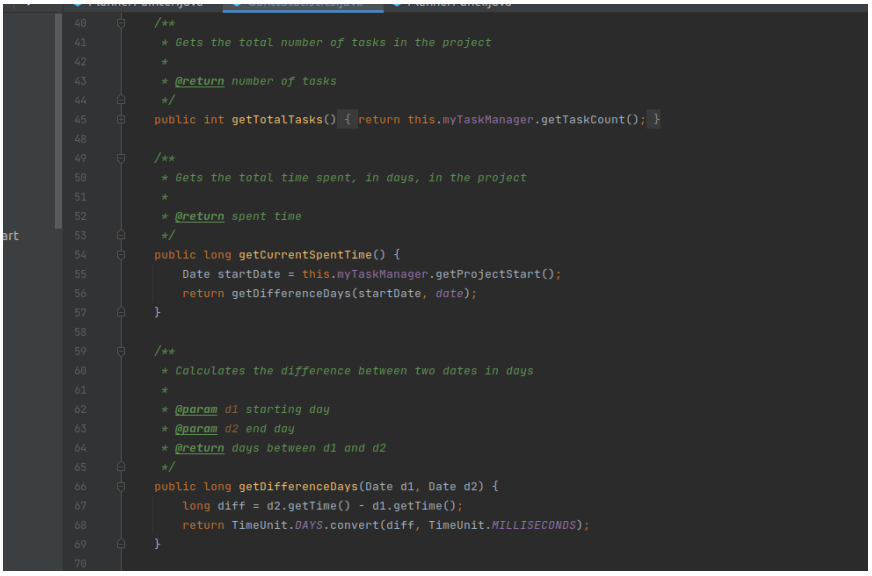
Figura 2.2: Funções em ViewHolder que reduzem e retornam o índice da view

```
public Object getSelectedUserObject() {
  Object selectedComp = this.getSelectedComponent();
  return myUserObjectsMap.get(selectedComp);
}
/**
 * Returns a user object with a given component (key)
 *
 * @param component key
 * @return object
 */
public Object getUserObjectAt(Component component){
  return myUserObjectsMap.get(component);
}
}
```

Figura 2.3: Função adicionada em GanttTabbedPane para retornar um objeto do tipo Object do mapa myUserObjectsMap.

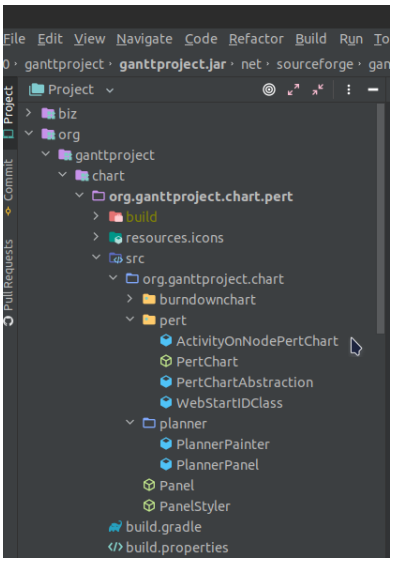
Novas funcionalidades

Durante o planeamento do projeto, foi notado que ambas as funcionalidades necessitam de ter uma forma de receber as estatísticas necessárias, tanto para as mostrar numericamente e sob a forma de uma barra de progresso no *Planner*, como para desenhar corretamente os gráficos no *Burndown Chart*. No seguimento desta discussão, foi visto como essencial a criação de uma classe que satisfizesse estes pedidos. Esta classe chama-se *GanttStatistics*, e esta é responsável por fornecer corretamente todas as métricas essenciais para o funcionamento dos dois painéis. Esta classe encontra-se na diretoria principal do *GanttProject*, ganttproject, dentro da pasta *net.sourceforge.ganttproject*



```
/**
 * Gets the total number of tasks in the project
 *
 * @return number of tasks
 */
public int getTotalTasks() { return this.myTaskManager.getTaskCount(); }

/**
 * Gets the total time spent, in days, in the project
 *
 * @return spent time
 */
public long getCurrentSpentTime() {
    Date startDate = this.myTaskManager.getProjectStart();
    return getDifferenceDays(startDate, date);
}

/**
 * Calculates the difference between two dates in days
 *
 * @param d1 starting day
 * @param d2 end day
 * @return days between d1 and d2
 */
public long getDifferenceDays(Date d1, Date d2) {
    long diff = d2.getTime() - d1.getTime();
    return TimeUnit.DAYS.convert(diff, TimeUnit.MILLISECONDS);
}
```

Figura 1: Exemplo de código produzido na classe GanttStatistics



Todo o restante código produzido para as duas novas ferramentas encontra-se no módulo *org.ganttproject.chart.pert*, em *org.ganttproject.chart*. Várias tentativas de mudança de nome foram realizadas, nenhuma com êxito. Observou-se que existiam dependências nos ficheiros de *build* no *Gradle* que recriam novamente o módulo, após o mesmo ser renomeado. Para adicionar, o gestor de *Plug-ins* também deixava de operar como desejado. Desta forma, foi decidido ignorar o nome do módulo, de modo a continuar com a implementação das funcionalidades.

Figura 2: Diretoria do código sobre as funcionalidades

Planner

O *Planner* é uma funcionalidade que permite visualizar algumas estatísticas de projetos criados na aplicação GanttProject. As estatísticas apresentadas por esta ferramenta têm o intuito de colocar o utilizador informado e atualizado intuitivamente sobre o projeto que está a criar. Foi decidido que esta funcionalidade seria implementada a partir de um novo painel, semelhante aos painéis *Gantt* e *Resources Chart*.

Com esta funcionalidade é possível visualizar as seguintes métricas: número total de tarefas, tempo gasto no trabalho (em dias), tempo total estimado para o trabalho (em dias), número total de tarefas já acabadas e o progresso total. Para os tempos, em dias, não são ignorados os fins de semana, e para o progresso total, é também considerado o progresso de cada tarefa.

Para aceder a este painel (ou *view*), basta clicar no menu *View*, na barra superior, onde aparecerá um botão com nome *Planner*. Ao clicar aqui, aparecerá o novo painel, como mostrado nas seguintes imagens.

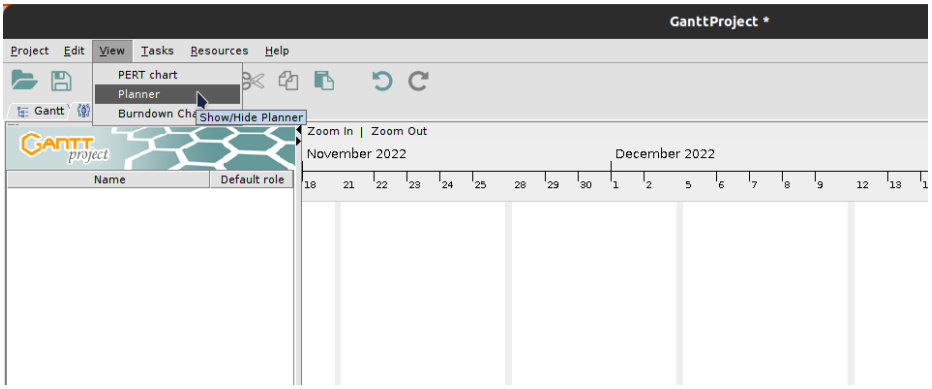


Figura 1: Botão para abrir o painel

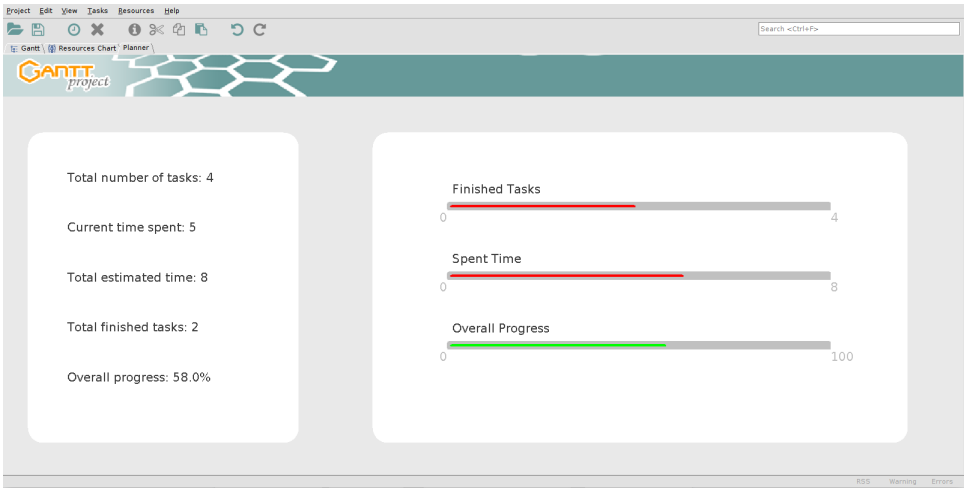


Figura 2: O painel aberto num projeto com algumas tarefas

Como é possível observar, esta *view* contém duas secções. Uma na esquerda, onde são apresentadas as estatísticas na sua totalidade e a nível numérico; e outra à direita, que apresenta as mesmas num formato de barras de progresso. A barra *Finished Tasks* apresenta o rácio entre tarefas terminadas e o total de tarefas; a barra *Spent Time* apresenta o rácio entre o tempo gasto e o tempo estimado, e por último, a barra *Overall Progress* apresenta o progresso total.

Nenhum botão na barra de ferramentas (ou *Toolbar*) tem utilidade neste painel, uma vez que este é estático e não permite editar informações.

É permitido a exportação do seu conteúdo para uma imagem. As seguintes imagens explicam como:

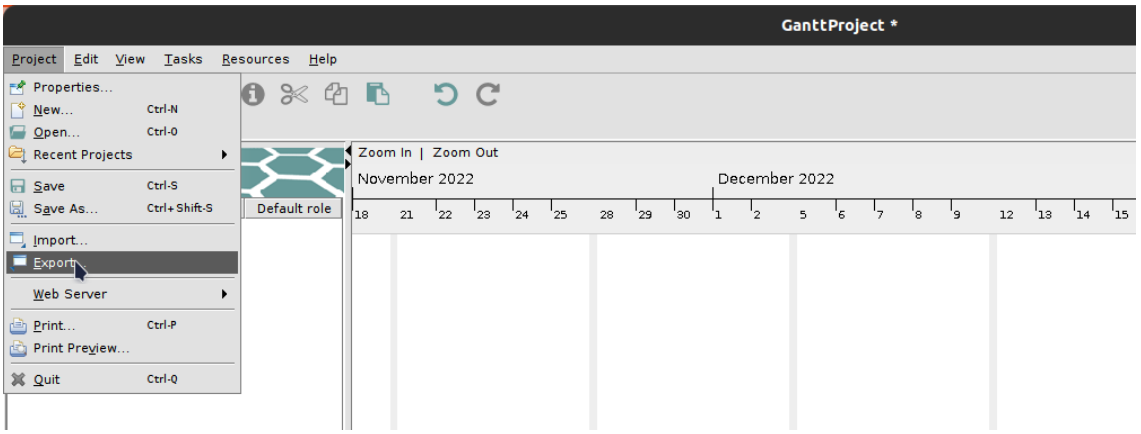


Figura 1: Passo 1. Clicar em export, no menu Project

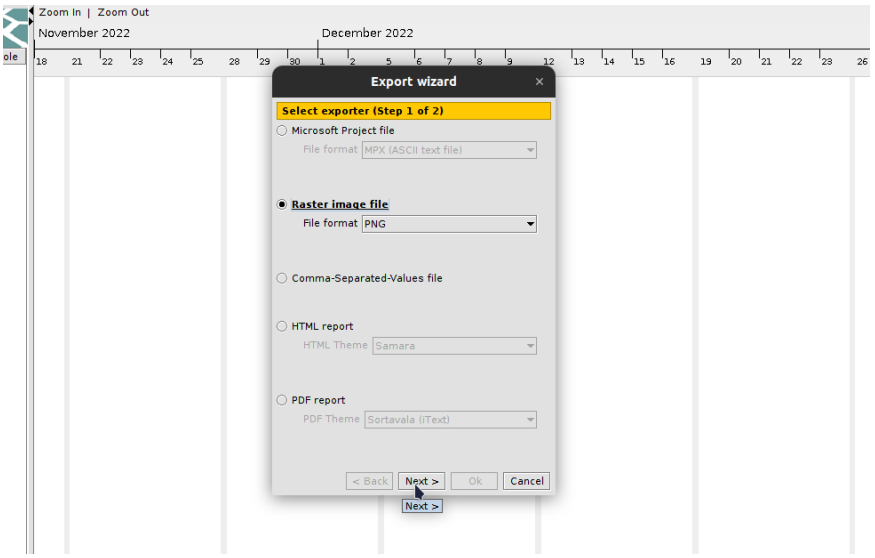


Figura 2: Passo 2. Selecionar Raster Image file, escolher formato e clicar em Next

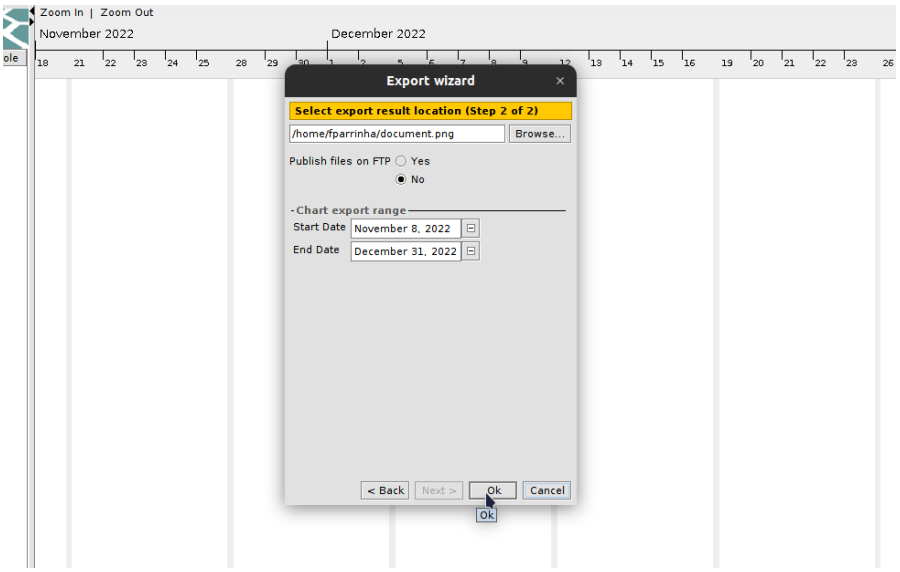


Figura 3: Escolher diretoria e clicar em Ok

Caso o projeto já esteja acabado, o painel, na secção com as barras de progressão, é ajustado de modo a apresentar um texto, a verde, com a indicação de que o projeto já foi concluído. Caso o projeto tenha acabado num tempo passado, o número de dias passados é apresentado. Se o trabalho foi concluído antecipadamente, o projeto apresenta o número de dias que faltam até a data estimada para a sua conclusão.

Contrariamente, caso o projeto esteja atrasado, o texto é apresentado na cor vermelha, com a indicação do número de dias em atraso.

Figura 4: Exemplo de projeto acabado mais cedo

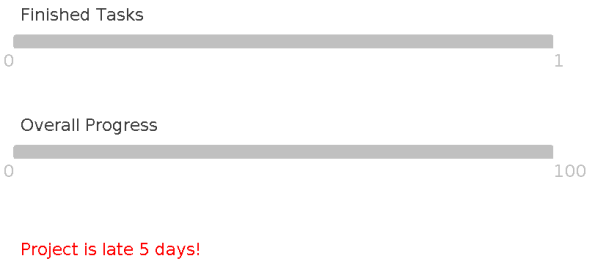


Figura 5: Exemplo de projeto atrasado

Desafios

A implementação do Planner apresentou alguns desafios. O principal foi entender como integrar com o código já presente. Foram realizadas duas tentativas e só na segunda, ao se descobrir da existência do gestor de *Plug-ins*, é que o seu desenvolvimento ocorreu de forma mais linear. Seguidamente, foi necessário entender como o *PERT Chart* estava implementado, de modo a criar a nova extensão. Esta parte obrigou a um estudo mais aprofundado da estrutura do código do GanttProject, que apresentou alguns problemas, especialmente relacionados com a reutilização e seleção do código já existente. Estes passos aceleram imenso o desenvolvimento da segunda funcionalidade.

De resto, a criação da componente gráfica foi mais fácil, no entanto, mesmo assim, foi necessário um estudo sobre a biblioteca gráfica *Swing*.

Burndown Chart

Por forma a tornar completa a experiência com o GanttProject, como um meio de controlo de projectos para além do anteriormente descrito, foi decidido implementar um *plug-in* que representa um burndown chart. Esta é uma representação gráfica do trabalho a ser feito em função do tempo. O trabalho pendente (ou *backlog*) está representado no eixo vertical (tempo de trabalho medido em dias e sem contabilização de fins de semana) e o tempo total estimado do projeto ao longo do eixo horizontal (contabilizando fins-de-semana ajustando assim a progressão temporal a um contexto real) sendo útil para prever quando todo o trabalho será concluído. É frequentemente usado em metodologias ágeis de desenvolvimento de software, como *Scrum*.

São desenhadas três curvas: uma, a vermelho, para a estimativa; outra, a azul, para o número de tarefas que restam; e por último, uma a verde, para representar o trabalho por realizar. A curva vermelha, de estimativa, é sempre uma reta para representar o *burn* ideal.

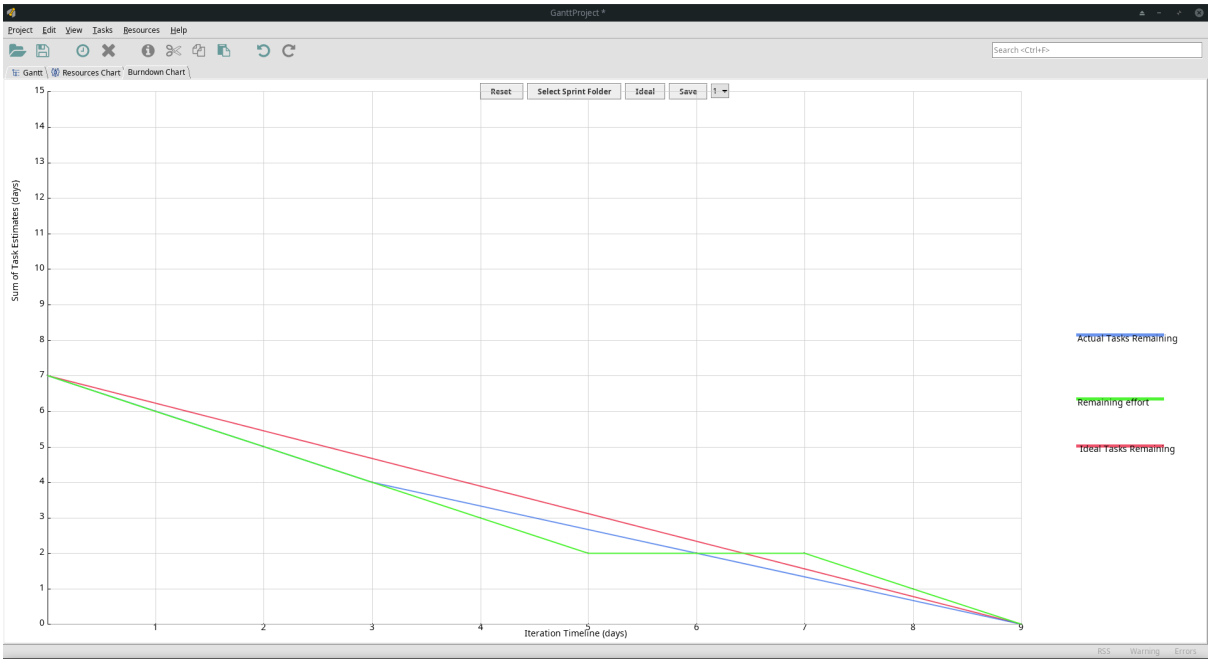


Figura 1: Grafico com as 3 linhas com cores diferentes

Modo de operação

Foi decidido implementar dois modos de operação, um modo ***Ideal*** e um modo ***History***:

- No modo ***Ideal***, o plugin desenha as restantes curvas desde o início do projeto até ao estado atual das tarefas como se tudo tivesse sido feito na perfeição, sem ter em conta a data de hoje, apenas um gráfico de como seria o perfeito burndown se todas as tarefas fossem feitas linearmente ao longo do seu tempo. Este modo oferece uma estimativa de como será o *burndown chart* real para qualquer progressão escolhida.

- No modo ***History*** introduziram-se maneiras de ir acompanhando os estados antigos do projeto, permitindo desenhar um gráfico de como realmente foi o desenvolvimento do projeto.

Para o modo ***History*** foram adicionados três botões e uma *combobox*, com o propósito de gerir os logs dos estados anteriores do projeto:

- Através do botão "*Select Sprint Folde*r" o utilizador pode selecionar uma pasta que irá conter os logs do estado do projeto em cada dia do sprint, que o utilizador grave, como mostrado na figura 1.
- Através do botão "*Reset*" são apagados os logs de todos os dias do sprint.
- Através do botão "*Save*" é guardado o estado atual das tarefas no dia (relativo ao início do projeto) escrito pelo utilizador na *combobox* adjacente.

Pré-condição: É importante notar que, para o bom funcionamento deste modo, a diretoria tem que obrigatoriamente corresponder ao *sprint* correto, caso contrário, o gráfico receberá informações incorretas, fazendo uma má pintura das curvas.

Nota: O botão que troca entre os modos muda de modo quando clicado, mas o gráfico só é redesenhado na próxima vez que o utilizador entrar no tab.

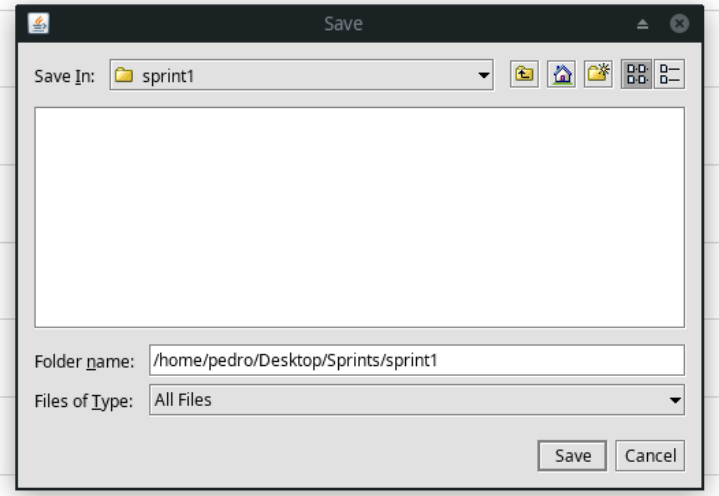


Figura 2: Escolher diretoria e clicar em Save

Esta implementação suporta lacunas nos ficheiros, ou seja, se o utilizador por acaso quiser fazer logs so de x em x dias do estado do projeto, ou até caso se esqueça num determinado dia de o fazer, o plugin desenha apenas os pontos introduzidos unidos por linhas. A maneira como esta *feature* foi implementada tem as suas limitações, os ficheiros guardam para cada task a sua duração e percentagem, levando consigo maneiras de representar cenários específicos:

- Quando se dá *resize* em tasks previamente guardadas, o plugin não desenha os pontos anteriores tendo em conta o novo tamanho, ou seja se inicialmente se introduziu uma task de 4 dias a 50% e no dia seguinte deu-se *resize* dessa task para 8 dias e nao se mudou os 50%, o ponto anterior vai ter em conta 2 dias feitos da task e o ponto corrente 4 dias. Foi decidido ser implementado assim de maneira a ficar mais preciso, guardar para cada ponto o verdadeiro estado atual, se o tempo total muda no estado dos pontos seguintes simplesmente empurra os estados anteriores exatamente como estavam para cima ou para baixo no gráfico.

- Quando os *resizes* das tarefas do projeto fazem com que a soma dos dias de todas as tarefas seja menor que os dias já feitos do ponto anterior, o plugin desenha pontos negativos no gráfico, não suportados. Este cenário foi considerado como um problema de utilização e não da implementação, tal como possíveis desformatações que desenhem linhas do gráfico fora dos limites. Isto porque o burndown chart é suposto representar quantos dias desde o tamanho total do projeto já foram trabalhados, e é exatamente isso que faz, se por acaso o utilizador num certo dia guarda um *log* com os dias já feitos e noutro dia dá resize às tasks de tal maneira

que muda o total número de dias, se o novo número total de dias for menor/maior do feito anteriormente é expectável que o gráfico desenhe valores fora dos limites.

Acessos

Decisões de visuais e de acesso de implementação foram tomadas, nomeadamente:

- Para aceder ao painel (ou *view*), mantém-se o critério do *Planner* bastando clicar no menu *View*, na barra superior, onde aparecerá um botão com nome *Burndown Chart*. Ao clicar aqui, aparecerá o novo painel, como mostrado na seguinte imagem:

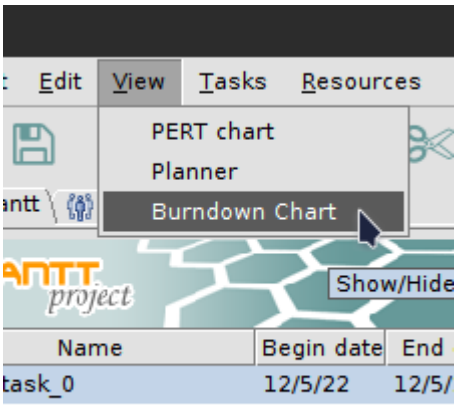


Figura 3: Escolher Burndown Chart para adicionar o tab

- Foram ainda adicionadas algumas funcionalidades extra tais como a possibilidade do *resize* da window sem deformação do gráfico e talvez mais relevante, a possibilidade de exportação do *Burndown* para um ficheiro de imagem raster como por exemplo .JPEG e .PNG. A exportação do gráfico realiza-se do mesmo modo que no planner.

Desafios

Os principais desafios encontrados no desenvolvimento desta componente acabam por ser muito idênticos com os já identificados anteriormente na componente *Planner*, nomeadamente a integração do código já existente. No entanto, é de realçar que o facto de se ter avançado primeiramente na implementação do *Planner*, facilitou consideravelmente todo o progresso de desenvolvimento do *Burndown Chart*.

Use Cases

Para o projeto foi produzido um *use case* que descreve a utilização das funcionalidades implementadas.

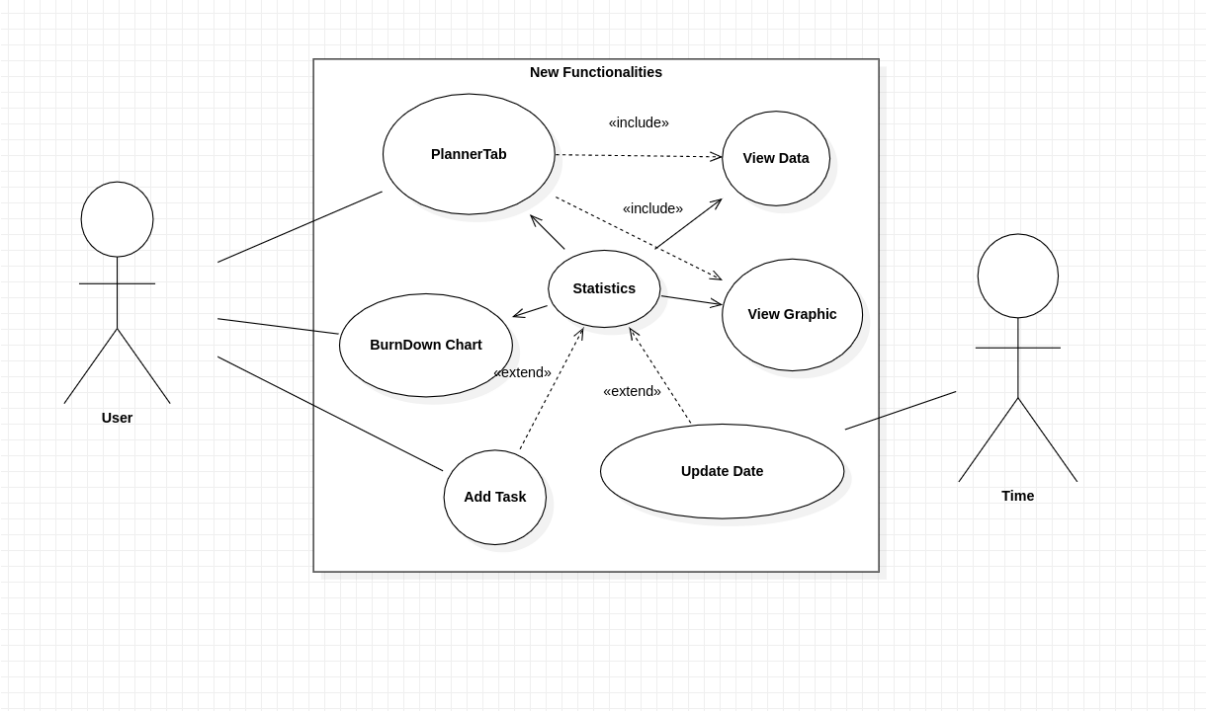


Figura 1: Use case de toda a implementação (as duas funcionalidades).

Nota: A ideia inicial para a atual divisão do relatório foi de apresentar dois *use cases* separados. Um para o *Burndown Chart* e outro para o *Planner*. No entanto, durante o desenvolvimento de ambos, foi notado que algumas classes seriam reutilizáveis. Neste seguimento, foi decidido juntar ambos os use cases num só, demonstrando a reutilização do código, nomeando o novo para *New Functionalities*.

Todos os use cases produzidos individualmente encontram-se nas diretorias pedidas pelo enunciado, na pasta do projeto.

Use Case Description

UML Id: Implemented Code
Name: New Functionalities

Description: This tab shows a window with the statistics of the current project alongside with a graphical representation of those and also a burndown chart of the project based on the statistics in another tab. The user can add or complete a task to change this data and update the statistics class.

User Story

UML Id: Implemented Code
User Story: Hi! I'm Paulo. Looking at the Gantt project I feel a lack of clear ways to see the overall progress and status of my project. I would also like to be able to see my progress alongside the estimated progress and see if I'm fulfilling the expectations of this project or not.

Notas

Nesta seccção serão discutidos alguns pontos adicionais não diretamente relacionados com os tópicos previamente mencionados.

No histórico *git* existe um perfil com nome “--add”.. Este perfil corresponde ao aluno Francisco Parrinha, número 58360. Este nome surgiu por erro ao realizar alguns comandos no terminal relacionados com o git. O problema só foi identificado mais tarde, resultando num grande número de *commits* identificados com “–add”.

Como foi dito na introdução, é importante realizar os passos relacionados com o *git remote*, de modo a obter a totalidade das *braches* do projeto. Sem estes passos, o *build* do projeto resulta numa falha durante a execução de um comando *git*. Assim sendo é essencial realizar os passos mencionados na introdução para que a compilação corra com sucesso. Esta necessidade surgiu uma vez que foi decidido não criar um novo repositório, como indicado no segundo guia, de modo a preservar o trabalho realizado na fase um.

Apesar de terem sido realizados apenas dois *sprints*, todas as semanas ocorreram reuniões matinais, às segunda-feiras, com o intuito de discutir principalmente as funcionalidades a serem implementadas, assim como corrigir os problemas iniciais relacionados com o *build* do projeto. O *log* das reuniões encontra-se escrito em papel, assim como num ficheiro [txt] no repositório git.

Conclusão

Em resumo, verificamos que este projeto foi marcante no nosso percurso como programadores. Foi a nossa primeira vez a contribuir para um projeto *open source*, especialmente um tão grande como este.

Desenvolvemos-nos como programadores desde o início ao fim. Tivemos inicialmente vários problemas de setup, o que nos desenvolveu as skills relacionadas com o *GIT*. O código do ganttProject que tivemos como base é confuso, não comentado e mal estruturado, foi no mínimo um desafio perceber toda a estrutura do código e tentar implementar bem as funcionalidades, tendo até que corrigir um *bug* dentro do ganttProject para o funcionamento correto das nossas implementações. E até no fim, com o estudo das métricas, aprendemos e aprofundamos conhecimentos que qualquer engenheiro informático deveria ter.

Acreditamos que os nossos resultados neste projeto tenham alcançado as nossas expectativas, e ter participado e sucedido num projeto tão grande e desafiante como o GanttProject desenvolveu das nossas capacidades como programadores. Deu-nos confiança para no futuro contribuir para outros projetos *open source*.