INE 5404 - POO II - Projeto Final

CheckSpace

Quenio Cesar M. dos Santos

1 de dezembro de 2014

CheckSpace

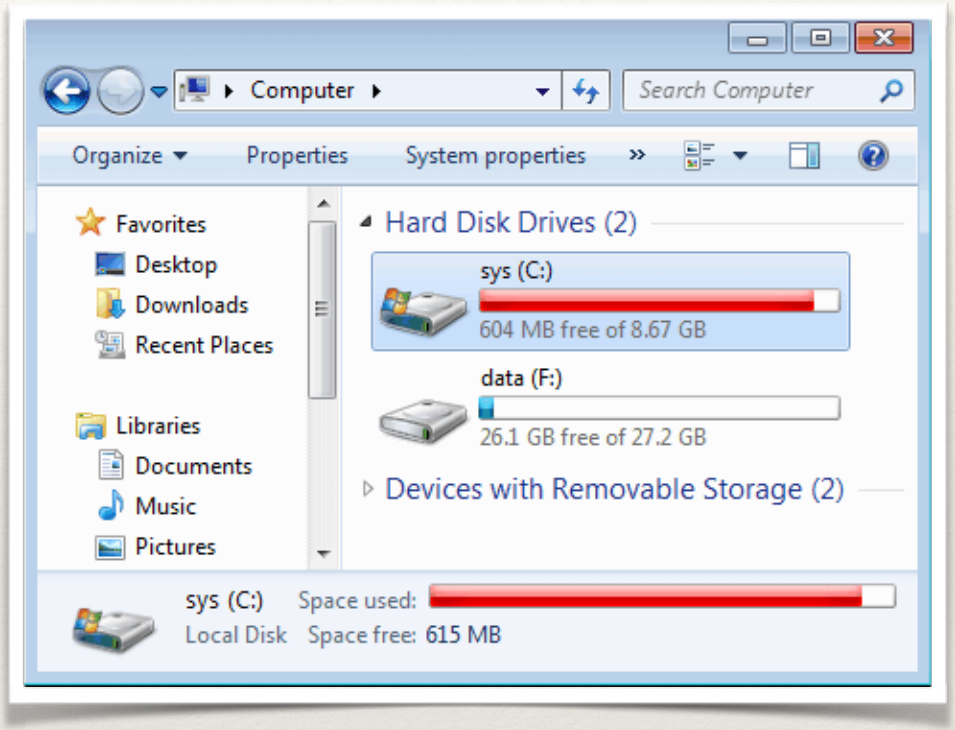
Propósito

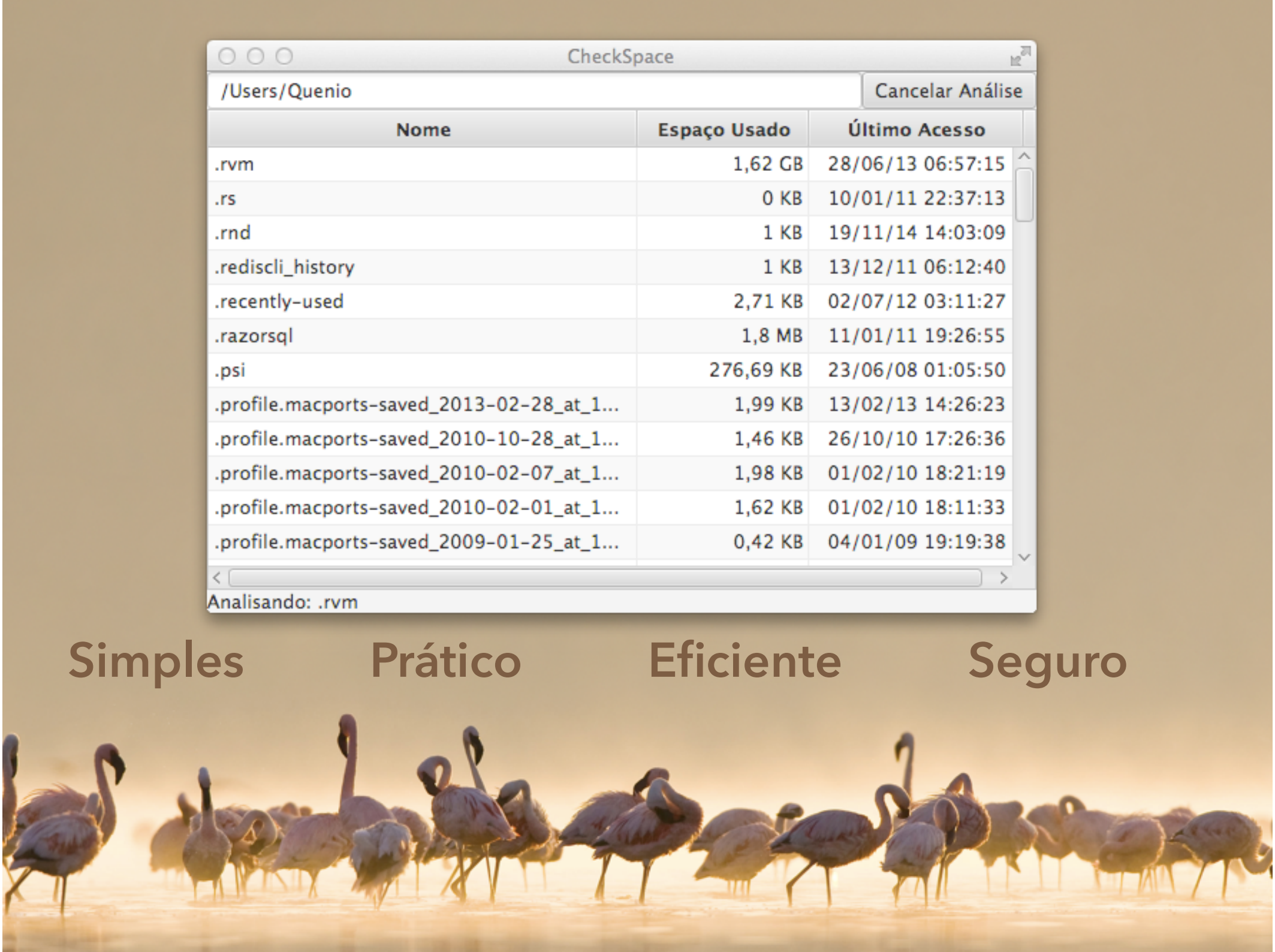
- Auxiliar o usuário na liberação de espaço em seu computador.
- Informa quais são as pastas que podem ser removidas ou arquivadas sem prejudicar o fluxo de trabalho.

CheckSpace

O Espaço Acabou

- Muitas fotos, vídeos, música, documentos, etc.
- O que fazer quando o espaço acabou?

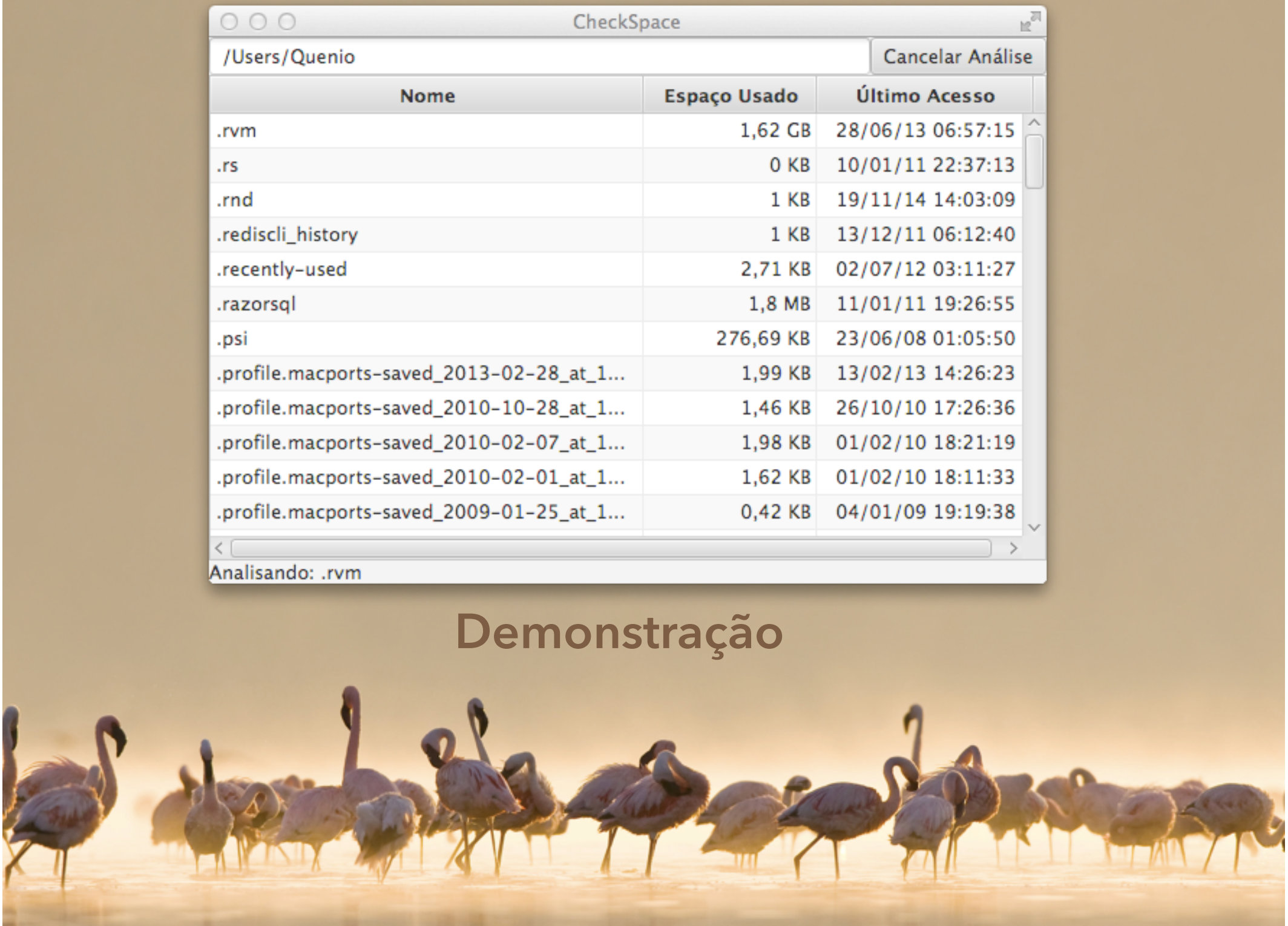



CheckSpace
/Users/Quenio
Cancelar Análise
Nome
Espaço Usado
Último Acesso
.rvm 1,62 GB 28/06/13 06:57:15
.rs 0 KB 10/01/11 22:37:13
.rnd 1 KB 19/11/14 14:03:09
.rediscli_history 1 KB 13/12/11 06:12:40
.recently-used 2,71 KB 02/07/12 03:11:27
.razorsql 1,8 MB 11/01/11 19:26:55
.psi 276,69 KB 23/06/08 01:05:50
.profile.macports-saved_2013-02-28_at_1... 1,99 KB 13/02/13 14:26:23
.profile.macports-saved_2010-10-28_at_1... 1,46 KB 26/10/10 17:26:36
.profile.macports-saved_2010-02-07_at_1... 1,98 KB 01/02/10 18:21:19
.profile.macports-saved_2010-02-01_at_1... 1,62 KB 01/02/10 18:11:33
.profile.macports-saved_2009-01-25_at_1... 0,42 KB 04/01/09 19:19:38
Analisando: .rvm
Simples
Prático
Eficiente
Seguro

CheckSpace

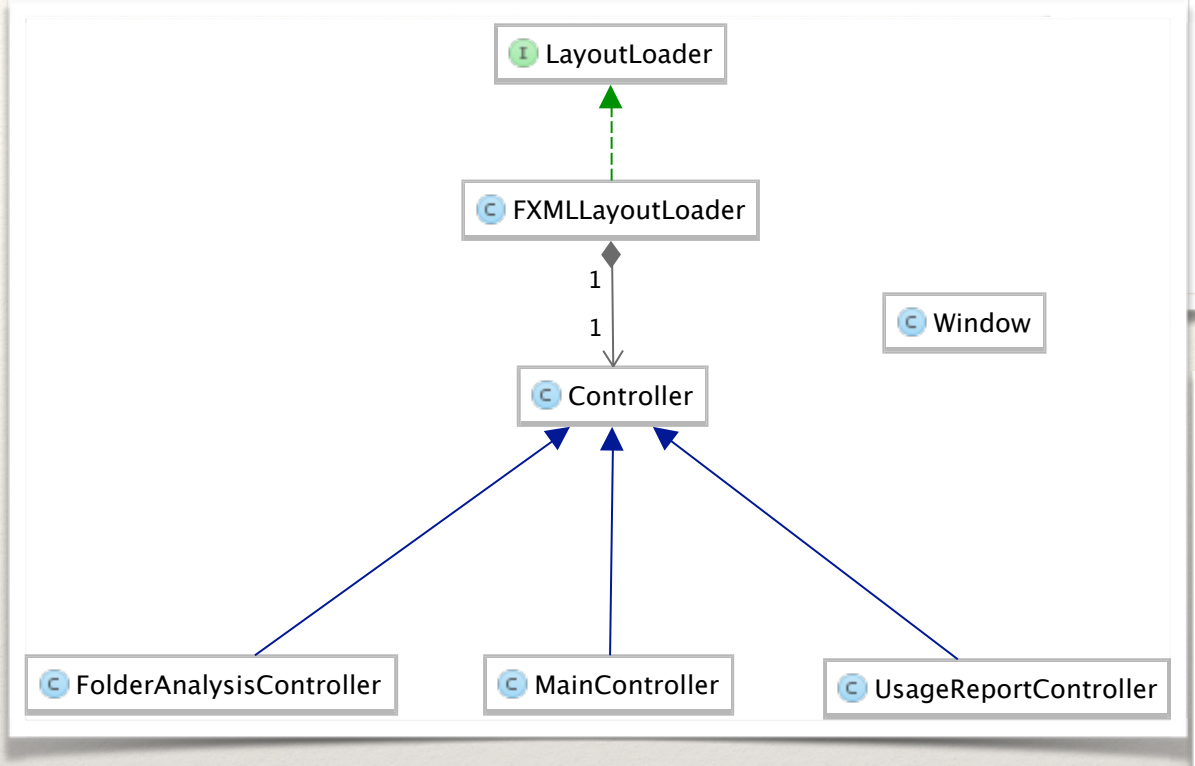
Análise de Pastas

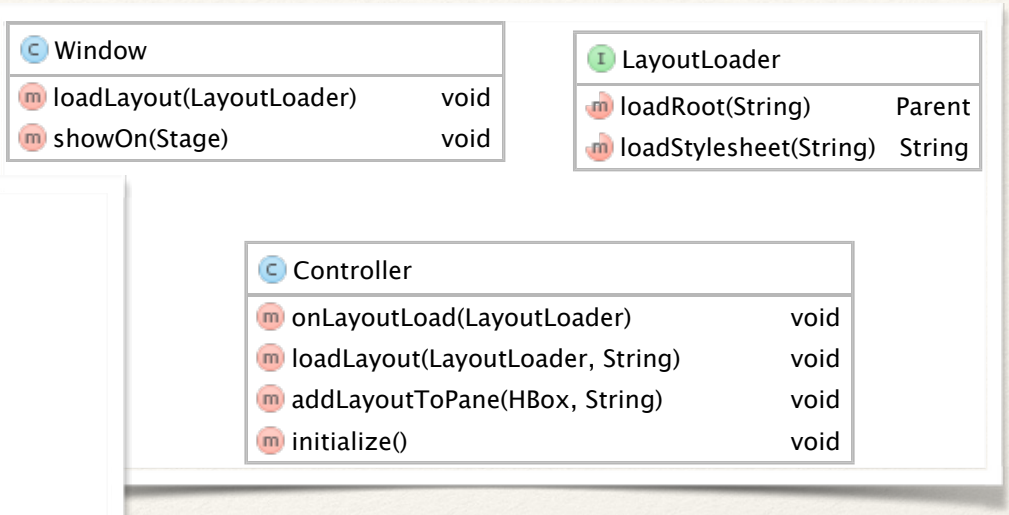
1. Digite o caminho da pasta
2. Pressione “Analisar Pasta”
3. O aplicativo analisa o conteúdo da pasta e fornece um relatório com:
   - Nome do arquivo ou sub-pasta
   - O espaço total ocupado
   - Último acesso


CheckSpace
/Users/Quenio
Cancelar Análise
Nome
Espaço Usado
Último Acesso
.rvm 1,62 GB 28/06/13 06:57:15
.rs 0 KB 10/01/11 22:37:13
.rnd 1 KB 19/11/14 14:03:09
.rediscli_history 1 KB 13/12/11 06:12:40
.recently-used 2,71 KB 02/07/12 03:11:27
.razorsql 1,8 MB 11/01/11 19:26:55
.psi 276,69 KB 23/06/08 01:05:50
.profile.macports-saved_2013-02-28_at_1... 1,99 KB 13/02/13 14:26:23
.profile.macports-saved_2010-10-28_at_1... 1,46 KB 26/10/10 17:26:36
.profile.macports-saved_2010-02-07_at_1... 1,98 KB 01/02/10 18:21:19
.profile.macports-saved_2010-02-01_at_1... 1,62 KB 01/02/10 18:11:33
.profile.macports-saved_2009-01-25_at_1... 0,42 KB 04/01/09 19:19:38
Analisando: .rvm
Demonstração

Arquitetura

- Model-View-Controller (MVC)
- Command Design Pattern
- Data/Property Binding
- Dependency Injection





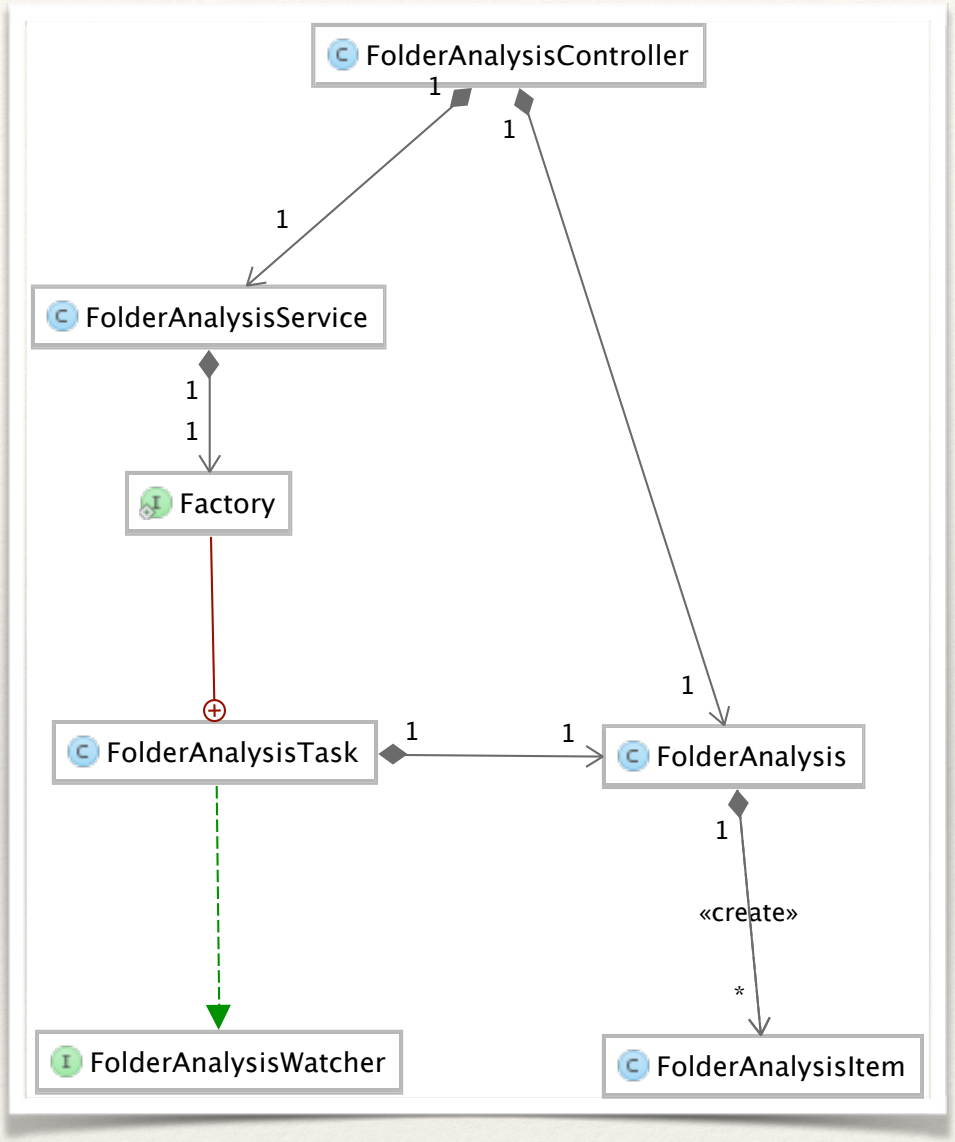

```
<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="3
    <children>
        <Button layoutX="126" layoutY="90" text="Click Me
        <Label layoutX="126" layoutY="120" minHeight="16"
    </children>
</AnchorPane>
```

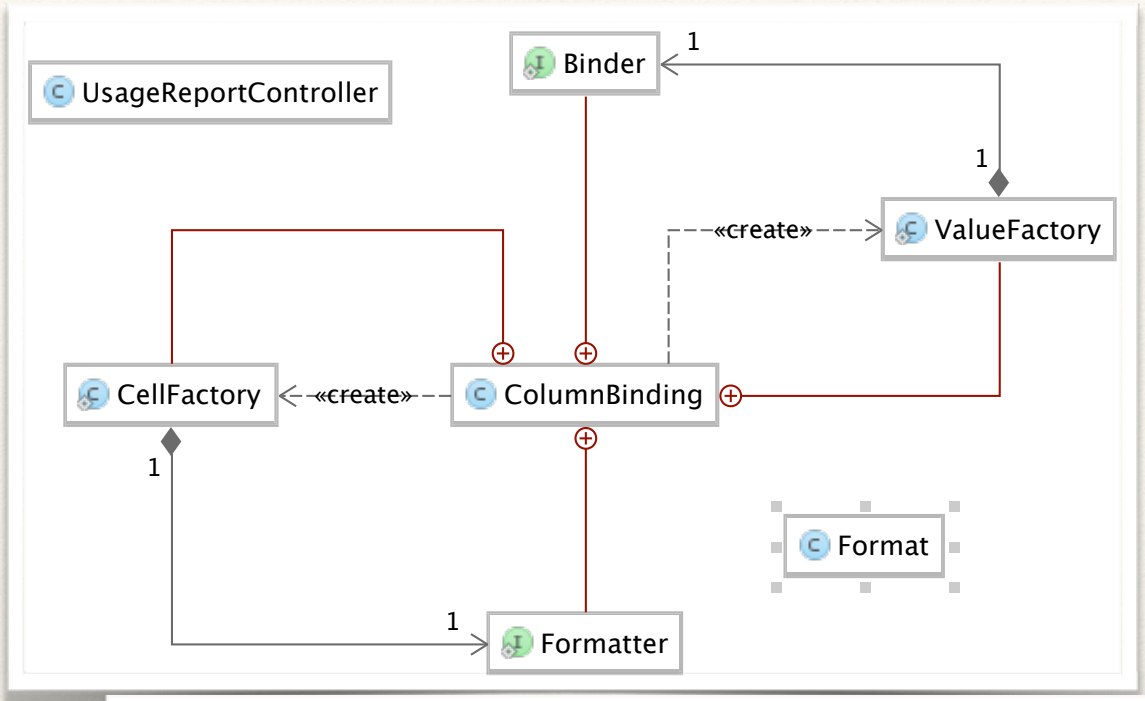

Interface Gráfica

- LayoutLoader carrega elementos visuais em Window.
- MainController utiliza LayoutLoader para montar Window.

Serviços e Tarefas

- *FolderAnalysisService* controla a execução de *FolderAnalysisTask*
- *FolderAnalysisTask* executa a análise num *background thread*
- *FolderAnalysis* faz a execução propriamente dita.

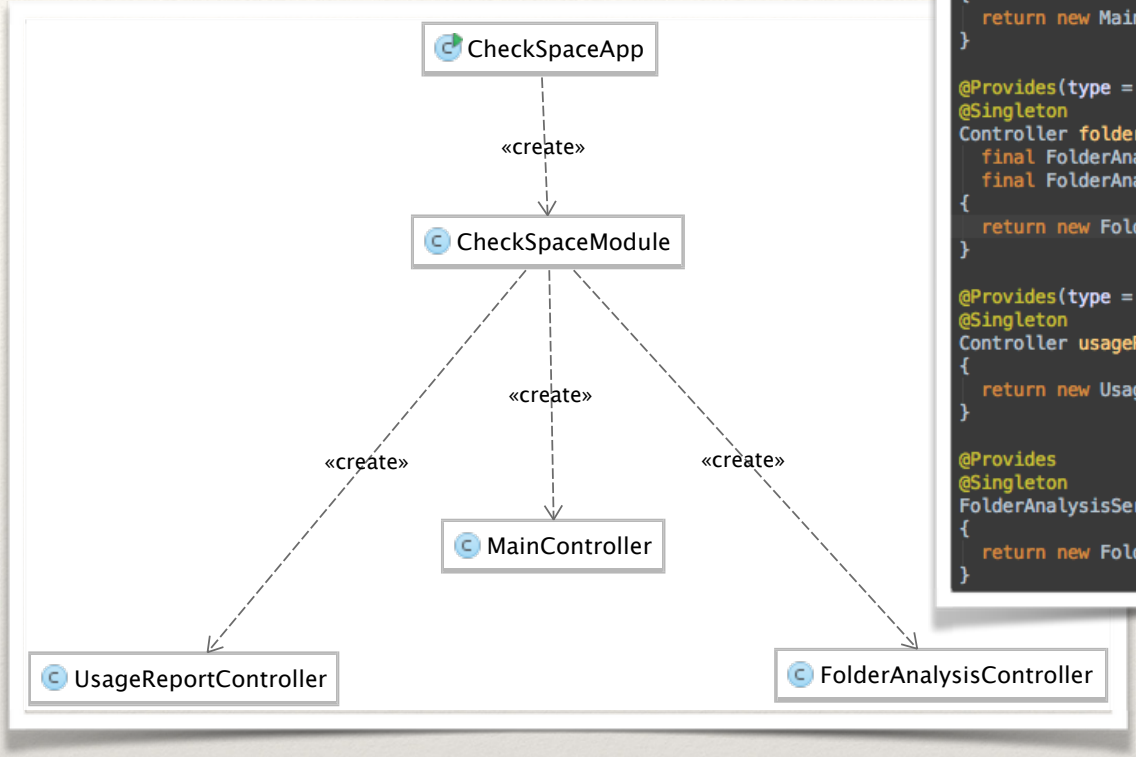





```
protected void initialize()
{
  folderAnalysis.bindListToItems(usageReport.getItems());
  ColumnBinding.of(nameColumn, FolderAnalysisItem::bindToName);
  ColumnBinding.of(spaceColumn, FolderAnalysisItem::bindToSpace, Format::space);
  ColumnBinding.of(lastAccessColumn, FolderAnalysisItem::bindToLastAccess, Format::date);
}
```

Relatório de Uso de Espaço

- *TableView* é configurada através de *property binding* pelo *UsageReportControlller*.




```java
@Singleton
Window mainWindow()
{
  return new Window(MAIN_WINDOW_RESOURCE_NAME, MAIN_WINDOW_TITLE, MAIN_WINDOW_WIDTH, MAIN_WINDOW_HEIGHT);
}

@Provides(type = Provides.Type.SET)
@Singleton
Controller mainController(final FolderAnalysisService folderAnalysisService)
{
  return new MainController(folderAnalysisService);
}

@Provides(type = Provides.Type.SET)
@Singleton
Controller folderAnalysisController(
  final FolderAnalysis folderAnalysis,
  final FolderAnalysisService folderAnalysisService)
{
  return new FolderAnalysisController(folderAnalysis, folderAnalysisService);
}

@Provides(type = Provides.Type.SET)
@Singleton
Controller usageReportController(final FolderAnalysis folderAnalysis)
{
  return new UsageReportController(folderAnalysis);
}

@Provides
@Singleton
FolderAnalysisService folderAnalysisService(final FolderAnalysisTask.Factory taskFactory)
{
  return new FolderAnalysisService(taskFactory);
}
```


Grafo de Objetos

- *Dagger* constrói automaticamente o grafo de todos os objetos.
- *CheckSpaceModule* permite verificar toda a árvore de dependência.

Conclusão

- Arquitetura baseada nos princípios, padrões de projeto (*MVC, Command, DI*) e tecnologias (*Java FX* e *Dagger*)
- Evolução garantida - fácil adicionar mais funcionalidade sem mudanças estruturais.

Referências

- **Model-View-Controller:** http://en.wikipedia.org/wiki/model–view–controller
- **Command Design Pattern:** http://en.wikipedia.org/wiki/Command_pattern
- **Dagger:** http://square.github.io/dagger
- **Java FX:** http://docs.oracle.com/javase/8/javase-clienttechnologies.htm