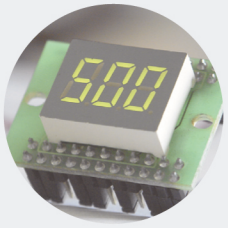
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Desenvolvimento de Aplicação para Visualização de Resultados Paramétricos de Wafers

JOSÉ PEDRO CHAIRA OLIVEIRA E CUNHA
Julho de 2023

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Desenvolvimento de Aplicação para Visualização de Resultados Paramétricos de Wafers

José Pedro Chaira Oliveira e Cunha

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Julho, 2023

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: José Pedro Chaira Oliveira e Cunha, Nº 1161530,
1161530@isep.ipp.pt

Orientação Científica: Carlos José Ribeiro Campos, crc@isep.ipp.pt

Empresa: Amkor Technology Portugal

Orientador: Filipe Matos, filipe.matos@amkor.com

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Julho, 2023

A chegar ao fim de mais uma etapa importante na minha vida académica e depois de ultrapassados vários obstáculos, gostaria de deixar alguns agradecimentos às pessoas que fizeram parte deste percurso. Primeiramente, gostaria de agradecer à minha família por me ter dado todo o apoio e a oportunidade poder de seguir o que mais gosto de fazer. Gostaria de agradecer ao meu orientador do ISEP, Engenheiro Carlos José Ribeiro Campos e também ao meu orientador da Amkor Technology Portugal, Engenheiro Filipe Matos. Estiveram sempre disponíveis tanto presencial como virtualmente para as minhas questões. Agradeço a sua disponibilidade e generosidade em guiar-me nesta nova aventura que foi a realização da tese.

Resumo

A evolução da tecnologia tem contribuído para o desenvolvimento da sociedade e os semicondutores surgiram num contexto onde é possível produzir dispositivos eletrónicos cada vez mais pequenos e complexos. No processo produtivo de semicondutores, pode-se identificar as seguintes etapas: Crescimento do cristal de silício, corte e laminação de discos designados por *wafers*, limpeza para remover contaminantes, dopagem para modificar as propriedades elétricas, litografia para definir padrões dos circuitos, gravura para remover as camadas indesejadas, implantação iónica para ajustar as características elétricas e corte da *wafer*.

No processo de fabrico de semicondutores, existem tarefas que têm de ser realizadas por engenheiros antes de seguirem para o cliente, como a análise de dados paramétricos obtidos nos teste das *wafers*. A análise, por norma, é feita através de *wafermaps*, que são representações gráficas usadas na indústria de semicondutores para visualizar e analisar resultados paramétricos, fornecendo uma representação visual dos *chips* individuais, chamados *dies*. Esse processo de análise manual por um engenheiro, onde tem que abrir um ficheiro de resultados, selecionar os parâmetros a analisar, introduzi-los num gráfico e depois efetuar um estudo analítico, pode-se tornar uma tarefa morosa e dispendiosa de recursos.

Este trabalho aborda o desenvolvimento de uma aplicação que será usada na etapa de análise da dados paramétricos. O *software* tem como principal objetivo ajudar o engenheiro na deteção de falhas de fabrico de *wafers*. No processo de desenvolvimento da aplicação, foram realizadas várias iterações com a equipa de engenheiros, até que esta fosse de encontro ao pretendido pela equipa.

Ao longo de todo o processo de criação da interface, todas as funcionalidades foram validades de forma a comprovar o seu correto funcionamento. Para avaliar a solução final, foi realizado um inquérito de usabilidade aos utilizadores da aplicação, a fim de compreender se a aplicação desenvolvida ia de encontro aos objetivos dos utilizadores da mesma.

Por fim, pode-se concluir que a aplicação desenvolvida permite efetuar uma análise de dados paramétricos através do estudo de um gradiente de cores no formato de *wafermap*.

Palavras-Chave: Semicondutores, Aplicação, *Wafer*, Gradiente de Cores.

Abstract

The evolution of technology has contributed to the development of society, and semiconductors have emerged in a context where it is possible to produce increasingly smaller and more complex electronic devices. In the semiconductor production process, the following stages can be identified: Growth of the silicon crystal, cutting and slicing of discs called wafers, cleaning to remove contaminants, doping to modify electrical properties, lithography to define circuit patterns, etching to remove unwanted layers, ion implantation to adjust electrical characteristics, and wafer cutting.

In the semiconductor manufacturing process, there are tasks that need to be performed by engineers before they are sent to the customer, such as analyzing parametric data obtained from wafer testing. Typically, this analysis is done using wafermaps, which are graphical representations used in the semiconductor industry to visualize and analyze parametric results, providing a visual representation of individual chips, called dies. This manual analysis process by an engineer, where they have to open a results file, select the parameters to analyze, input them into a graph, and then perform an analytical study, can become a time-consuming and resource-intensive task.

This work addresses the development of an application that will be used in the parametric data analysis stage. The software's main objective is to assist the engineer in detecting wafer manufacturing faults. During the application development process, several iterations were carried out with the team of engineers until it met the team's requirements.

Throughout the entire interface creation process, all functionalities were validated to ensure their correct functioning. To evaluate the final solution, a usability survey was conducted with the application's users to understand if the developed application met their objectives.

In conclusion, the developed application allows for the analysis of parametric data through the study of a color gradient in the wafermap format.

Keywords: Semiconductors, Application, Wafer, Colour Gradient.

Índice

Lista de Figuras **vii**

Glossário **ix**

Lista de Acrónimos **xi**

1 Introdução **1**
- 1.1 Contextualização . . . 2
- 1.2 Definição do Problema . . . 3
  - 1.2.1 Objetivos . . . 4
- 1.3 Plano de Trabalho . . . 5
- 1.4 Organização da Dissertação . . . 5

2 Estado de Arte **7**
- 2.1 Semicondutores . . . 7
  - 2.1.1 Wafers . . . 8
  - 2.1.2 Produção . . . 10
- 2.2 Linguagens de Programação . . . 13
  - 2.2.1 C# . . . 14
  - 2.2.2 Python . . . 15
  - 2.2.3 Java . . . 16
  - 2.2.4 Resumo . . . 18
- 2.3 Interface Gráfica do Utilizador . . . 18
  - 2.3.1 Windows Forms . . . 18
  - 2.3.2 Windows Presentation Foundation . . . 19
  - 2.3.3 Qt . . . 20
  - 2.3.4 GTK+ . . . 22
  - 2.3.5 Resumo . . . 23
- 2.4 Ambientes de Desenvolvimento Integrados . . . 24
  - 2.4.1 SharpDevelop . . . 24
  - 2.4.2 Visual Studio . . . 25
  - 2.4.3 Resumo . . . 26

3 Parte Prática **27**
3.1 Percentil . . . 28
3.2 Interface Gráfica . . . 30
3.3 Elementos da Interface . . . 31
3.3.1 Botões . . . 31
3.3.1.1 Botão Open File . . . 31
3.3.1.2 Botão Report File . . . 33
3.3.2 Etiquetas . . . 35
3.3.3 Caixas de Texto . . . 36
3.3.3.1 Caixa de Entrada de Texto . . . 36
3.3.3.2 Caixa de Entrada de Texto . . . 38
3.3.4 Listas . . . 41
3.3.4.1 Listas dos Números e Nomes dos Testes . . . 41
3.3.4.2 Lista de Parâmetros . . . 42
3.3.5 Tab Control . . . 43
3.3.6 Barras . . . 47
3.3.6.1 Barra Superior . . . 47
3.3.6.2 Barra Inferior . . . 48
3.4 Gestão de Dados em Memória . . . 50
3.4.1 Aquisição de Dados . . . 50
3.5 Correção e Prevenção de Erros . . . 53

4 Demonstração e análise de resultados **55**
4.1 Demonstração . . . 55
4.2 Apresentação das perguntas . . . 60
4.3 Análise de Dados . . . 62
4.3.1 Perguntas Gerais . . . 63
4.3.2 Perguntas Específicas . . . 64

5 Conclusões **71**
5.1 Trabalho Futuro . . . 72

Referências **74**

Anexo A Inquérito **79**

Lista de Figuras

1.1 Amkor Logótipo [1]. 3
1.2 Calendarização 5

2.1 Áreas de aplicações dos Semicondutores [5] 8
2.2 Tamanhos das *Wafers* [6]. 9
2.3 Exemplo de uma *wafer* [6]. 10
2.4 Cadeia de um Circuito Integrado [7]. 11
2.5 *Wafer fan-in vs Wafer fan-out* [8]. 12
2.6 Processo de *Recon* [7]. 12
2.7 C# [9]. 15
2.8 Python [9]. 16
2.9 Java [20]. 17
2.10 Exemplo de uma *GUI* desenvolvida em *Windows Forms* [23]. 19
2.11 Exemplo de uma *GUI* desenvolvida em *WPF* [26]. 20
2.12 Qt [27]. 21
2.13 Exemplo de uma *GUI* desenvolvida em Qt [29]. 21
2.14 GTK+ [27]. 22
2.15 Exemplo de uma *GUI* desenvolvida em GTK [32]. 23
2.16 SharpDevelop [35]. 24
2.17 *Visual Studio* [37]. 25

3.1 Visão geral da aplicação 28
3.2 Design do Percentil 28
3.3 Design do botão Open File 32
3.4 Design do botão *Report File* 33
3.5 *Labels* presentes na interface 35
3.6 *Design* das caixas de texto do percentil 36
3.7 Design das *Search Box* 38
3.8 *ListBox* Número e Nome dos Testes 41
3.9 *ListBox* de Parâmetros 42
3.10 Exemplo de um *wafermap* [1] 46
3.11 Menu *File* 48
3.12 Menu *Help* 48

3.13 Primeiro campo da barra inferior [2] 49
3.14 Segundo campo da barra inferior 49
3.15 Terceiro campo da barra inferior 49
3.16 Quarto campo da barra inferior 50

4.1 Interface inicial 56
4.2 Interface apenas com as unidades na *listbox* 57
4.3 Interface com unidades selecionadas 58
4.4 Interface com a *wafer* impressa no *TabControl* [3] 59
4.5 Informação presente no PDF gerado [4] 60
4.6 Interface da aplicação [5] 62
4.7 Perguntas Gerais 63
4.8 Interface com legenda [6] 64
4.9 Perguntas Específicas 65

Glossário

dicing

O processo de *dicing* de uma *wafer* é uma etapa importante na produção de circuitos integrados. Este processo envolve a separação dos *dies* individuais da wafer.

die

Os *dies* são os pequenos circuitos integrados que compõem uma *wafer*.

etching

O *etching* é um processo químico usado na microfabricação para remover seletivamente camadas finas de material de uma superfície, a fim de criar padrões complexos ou estruturas em escala micro ou nano.

fan-in

Uma *wafer fan-in* é uma *wafer* onde as ligações elétricas com o exterior apenas existem na área do circuito integrado.

fan-out

Uma *wafer fan-out* é uma *wafer* onde as ligações elétricas com o exterior existem na área do circuito integrado e também na sua periferia.

grinding

O processo de *grinding* de uma *wafer* é uma etapa importante na produção de circuitos integrados. Este processo envolve a remoção de uma pequena camada de silício da parte traseira da *wafer* para reduzir a sua espessura e torná-la mais uniforme.

mold-wafer

O processo de *mold-wafer* de uma *wafer* é uma etapa importante na produção de circuitos integrados. Neste processo, o *mold-wafer* é usado como suporte para moldar a resina *mold-compound*, que é um polímero líquido termoendurecível usado para encapsular os circuitos integrados.

Lista de Acrónimos

AS	Automação e Sistemas
ATEP	Amkor Technology Portugal
DEE	Departamento de Engenharia Electrotécnica
GTK	GIMP Toolkit
GUI	Graphical User Interface
IDEs	Integrated Development Environments
ISEP	Instituto Superior de Engenharia do Porto
JVM	Java Virtual Machine
MEEC	Mestrado em Engenharia Electrotécnica e Computadores
RDL	Redistribuição de Camadas Elétricas
SBA	Solder Ball Attach
TEDI	Tese/Dissertação
WPF	Windows Presentation Foundation
XAML	Extensible Application Markup Language

Capítulo 1

Introdução

Com o avançar dos tempos, a eletrónica tem vindo, cada vez mais, a revolucionar o mundo tecnológico e é neste contexto que existem os semicondutores. De uma forma resumida, um semicondutor é um material que possui uma condutividade elétrica intermediária entre os materiais condutores (materiais que permitem a passagem de corrente elétrica) e os materiais isolantes (materiais que não permitem a passagem de corrente elétrica).

Os semicondutores são materiais que possuem propriedades intermédias entre isolantes e condutores. A dopagem é um processo utilizado nos semicondutores para controlar a sua condutividade elétrica. Na dopagem tipo N, são adicionadas impurezas, como átomos de fósforo, que introduzem eletrões extra, aumentando a condutividade. Na dopagem tipo P, utiliza-se impurezas, como átomos de boro, que criam "lacunas"na estrutura, gerando portadores de carga positiva que também contribuem para a condução da corrente elétrica. Estes processos são essenciais na criação de componentes como díodos, transístores e *microchips*, permitindo o controlo da corrente elétrica e o funcionamento dos circuitos elétricos.

Desta forma, é possível ajustar as suas propriedades elétricas para torná-lo um bom condutor ou um bom isolante. Assim sendo, os semicondutores são então essenciais para a fabricação de dispositivos eletrónicos complexos.

1.1 Contextualização

Neste capítulo é apresentado o projeto desenvolvido durante o ano para a unidade curricular Tese/Dissertação (TEDI), do 2º ano do ramo de Automação e Sistemas (AS), do Mestrado em Engenharia Electrotécnica e Computadores (MEEC), do Departamento de Engenharia Electrotécnica (DEE), do Instituto Superior de Engenharia do Porto (ISEP). O projeto tem o seu principal foco no desenvolvimento de uma aplicação para visualização de resultados paramétricos de *wafers*, no processo produtivo de *microchips*.

No âmbito da unidade curricular TEDI realizou-se um estágio na empresa Amkor Technology Portugal (ATEP). A ATEP é uma subsidiária da Amkor Technology Inc, sendo um fornecedor de serviços de desenvolvimento, fabrico e teste de circuitos integrados para o mercado global. Fundada em 1968, a Amkor Technology Inc foi pioneira na sua área de atuação e é hoje um parceiro estratégico para mais de 250 das principais empresas a nível mundial.

A base operacional da Amkor inclui cerca de 1 milhão de metros quadrados de espaço com instalações de produção, centros de desenvolvimento de produtos, escritórios de vendas e suporte, localizados nas principais regiões de fabrico de eletrónica, na Ásia, Europa e Estados Unidos da América. Assim, a Amkor Technology, Inc. é um dos maiores fornecedores mundiais de serviços de *packaging* e teste de semicondutores.

A ATEP (Portugal) iniciou operações em 1996, sendo atualmente líder em tecnologias de *wafer Level Packaging* de 300mm, tanto *fan-in* como *fan-out*. A única diferença entre estes dois tipos de *wafers* encontra-se na área onde são feitas as ligações elétricas (conexões de entrada/saída). Nas *wafers fan-In*, as ligações elétricas com o exterior apenas existem na área do circuito integrado, enquanto que nas *wafers fan-Out*, as ligações elétricas com o exterior existem não só na área do circuito integrado, mas também na sua periferia.

Os produtos produzidos pela Amkor Technology, como referido anteriormente, são do tipo *fan-in* e *fan-out* sendo o seu processo dividido em várias etapas:

- Preparação das *wafers* (*waferprep*);
- *R*econstrução das *wafers* (*Recon*);
- Redistribuição de Camadas Elétricas (RDL);
- *Solder Ball Attach (SBA)*;
- Empacotamento dos circuitos integrados (*Packing*).

Situada em Vila do Conde, a ATEP suporta toda a cadeia de valor, desde o desenho até à implementação de múltiplas soluções tecnológicas que respondam às

necessidades mais exigentes dos clientes, servindo principalmente os mercados de telecomunicações, automóveis e segurança.

Figura 1.1: Amkor Logótipo [1].

1.2 Definição do Problema

Após o teste de cada *wafer* na linha de produção, é gerado um ficheiro com os valores paramétricos obtidos a partir dos testes realizados ao longo do processo. A análise dos valores obtidos dos diferentes parâmetros contidos nesses ficheiros, são cruciais para a deteção de possíveis erros no processo de fabrico até então, por isso, é imprescindível dedicar algum tempo à sua cuidadosa análise.

Para analisar esses dados, os engenheiros da ATEP utilizam o Microsoft Excel para conseguirem criar um gráfico no formato de *wafermap* com um gradiente de cores. Embora a utilização do Excel seja uma opção, o processo de seleção de um ficheiro pretendido, escolher os dados a analisar, criar um gráfico para desenhar o *wafermap* e, por fim, escolher as cores para o gradiente revela-se demorado, pouco eficiente e complexo. Trabalhar com esta ferramenta implica diversas etapas que consomem bastante tempo.

Primeiramente, é necessário escolher o ficheiro específico, após selecionar o ficheiro, é preciso identificar os dados específicos que se pretende analisar. Neste ponto, é necessário ter conhecimento sobre a estrutura do ficheiro e da localização dos dados desejados, o que pode requerer algum esforço para encontrar as informações pretendidas.

Uma vez selecionados os dados, é necessário criar um gráfico que represente o *wafermap*. Esta etapa implica a escolha de eixos adequados, a formatação correta dos dados e a configuração do gráfico para exibir as informações de forma clara e compreensível. Mais uma vez, isso requer familiaridade com as ferramentas e recursos do Excel, o que pode ser um processo complexo para aqueles menos familiarizados com a plataforma.

Por fim, é preciso escolher as cores para o gradiente do *wafermap*. Esta escolha não se trata apenas de uma questão estética por poder influenciar a interpretação dos dados.

Assim sendo, embora seja possível utilizar o Excel como uma solução para criar um *wafermap*, é importante reconhecer a sua baixa eficiente e a sua complexidade, o que acaba por se tornar num *bottleneck* na análise dos valores obtidos das respetivas *wafers*.

1.2.1 Objetivos

Este trabalho tem como principal objetivo desenvolver uma aplicação para visualização de resultados paramétricos de *wafers*.

Para a realização deste projeto tiveram-se em conta as seguintes tarefas:

- Analisar e perceber o formato dos resultados do teste em *wafer* na Amkor;
- Criar uma aplicação em C# que leia os dados a partir de ficheiros Excel;
- Criar um *Graphical User Interface (GUI)* que permita a seleção de um determinado parâmetro de teste;
- Apresentar os dados paramétricos no formato de *wafermap* com um gradiente de cores.

Com o objetivo de melhorar a aplicação foram implementadas funcionalidades adicionais, tais como:

- Implementação de duas *text box* para ser possível o utilizador escolher os valores dos percentis desejados;
- Implementação de uma barra superior com funcionalidades extra;
- Implementação de uma barra inferior para fornecer informações adicionais ao utilizador;
- Possibilidade de guardar as informações analisadas num ficheiro PDF;

Depois de efetuada uma pesquisa e estudo sobre o tema envolvente do projeto passou-se à fase de desenvolvimento da aplicação, tendo-se como objetivo atingir os seguintes resultados:

- Leitura e interpretação fácil dos dados paramétricos presentes nos ficheiros ZIP;
- Criação de uma interface *userfriendly* que permita ao utilizador escolher o teste que pretende analisar baseando-se nas unidades dos valores do teste;
- Apresentação dos dados paramétricos interpretados em formato de *wafermap*;
- Realização do relatório final do projeto.

1.3 Plano de Trabalho

Na figura seguinte está representado a calendarização do trabalho realizado ao longo de todo o projeto.

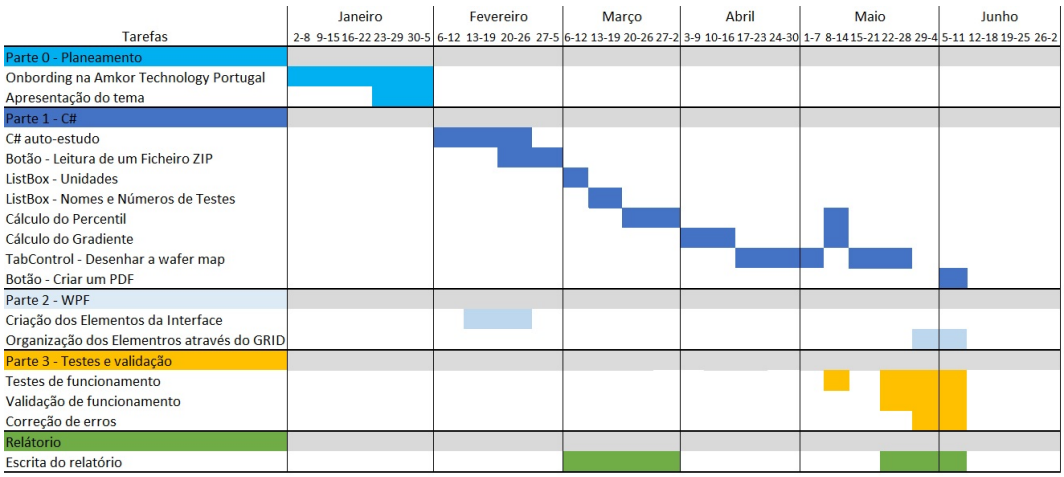


Figura 1.2: Calendarização

1.4 Organização da Dissertação

No Capítulo 1 é feita uma pequena contextualização do projeto, bem como a definição do problema e objetivos propostos. Já no Capítulo 2 é feito o estado de arte do projeto, evidenciando uma explicação sobre semicondutores, uma abordagem sobre as *wafers* e também a explicação de um processo de produção de semicondutores. Ainda neste capítulo é apresentado o estudo sobre formas de desenvolvimento de interfaces gráficas comparando-as, de forma a obter um maior conhecimento das mesmas. No Capítulo 3 é abordada toda a parte prática da dissertação, evidenciando o cálculo do percentil, a interface gráfica, os diferentes elementos da interface, a gestão de dados em memória e a correção e prevenção de erros. Já no Capítulo 4 é feita a demonstração do funcionamento da aplicação e também a análise de um inquérito criado para perceber a opinião dos utilizadores. Por fim, no Capítulo 5 é realizada uma conclusão acerca de todo o relatório.

Capítulo 2

Estado de Arte

Neste capítulo irá será apresentado o estado de arte das ferramentas, tecnologias e apresentados os conceitos relacionados com o projeto realizado na área dos semicondutores. Inicialmente, será apresentada uma abordagem aos semicondutores para que seja possível entender melhor a sua origem, função e a sua importância no mundo da eletrónica. Será depois abordado, neste capítulo, várias ferramentas existentes relativas à programação orientada a objetos, assim como linguagens de programação, de forma a estudar e entender qual será a melhor opção para a criação de uma interface gráfica interativa.

2.1 Semicondutores

A história dos semicondutores começou no final do século XIX, quando o físico alemão *Ferdinand Braun* descobriu o efeito retificador num cristal de sulfureto de chumbo. Na década de 1930, foram produzidos os primeiros semicondutores dopados. O processo de dopagem envolve a introdução de impurezas num material semicondutor, o que aumenta sua capacidade de condução de eletricidade [2]. Nos anos 40 foram desenvolvidos os primeiros transístores e o primeiro circuito integrado foi inventado no ano de 1958. Desde então, a tecnologia dos semicondutores tem sido fundamental para a evolução tecnológica [3] [4].

Os semicondutores são materiais sólidos de condutividade elétrica variável, podendo transitar de um material condutor para um material isolante com facilidade,

isto é, um semicondutor pode tornar-se um condutor ou isolador consoante as necessidades. Devido a esta alternância de condutividade, estes materiais estão presentes nos mais diversos aparelhos eletrónicos de alta tecnologia [4]. A seguir, apresenta-se uma figura que ilustra onde são utilizados os semicondutores no mundo da indústria:

Figura 2.1: Áreas de aplicações dos Semicondutores [5]

As propriedades versáteis e controladas que os semicondutores conseguem ter são fundamentais para os dispositivos eletrónicos complexos e é por este motivo que eles têm uma grande importância nos dias de hoje. No que toca à produção de semicondutores, atualmente, a grande maioria das indústrias recorrem a *wafers* que desempenham um papel fundamental na produção. Uma *wafer* é um disco onde estão desenhados inúmeros componentes eletrónicos, como transístores, díodos, ou até mesmo circuitos integrados complexos.

2.1.1 Wafers

Este processo de produção de semicondutores em *wafers* surgiu no final da década de 1950 pela empresa *Texas Instruments* e é usado até hoje pelo facto de permitirem que muitos *chips* sejam fabricados simultaneamente aumentando a eficácia e a redução de custos. As *wafers* nada mais são do que uns discos finos, circulares e planos de material semicondutor, geralmente de silício, que são usados como base para a fabricação dos semicondutores. Estas *wafers* são fabricadas para serem altamente planas e uniformes pois qualquer irregularidade pode afetar o desempenho dos componentes eletrónicos produzidos. Relativamente às suas dimensões, possuem uma variedade de diâmetros que vão desde os 25,4 milímetros até 450 milímetros, como se ilustra na Figura 2.2. As medidas mais comuns são as de 100, 150, 200 e 300 milímetros. Ao longo dos anos, o diâmetro foi aumentando pelo simples facto de se pretender reduzir os custos na produção. No que toca à sua espessura, esta depende do material que constitui a *wafer*. A espessura é medida pela resistência do material utilizado, ou seja, deve ser suficiente para suportar o seu próprio peso durante o manuseamento [6].

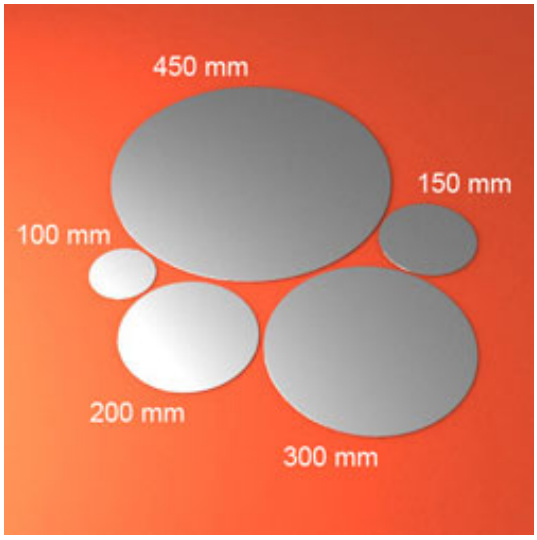


Figura 2.2: Tamanhos das *Wafers* [6].

De seguida, serão então apresentadas as vantagens da utilização das *wafers* como método de produção de semicondutores [5]:

- Controlo de Qualidade: As *wafers* são fabricadas sob condições extremamente controladas, para que não ocorram danos nos componentes. Isto é essencial para manter a uniformidade dos produtos e também para o controlo da qualidade dos mesmos;
- Custo-benefício: A utilização de *wafers* permite a produção em massa de semicondutores de alta qualidade, o que acaba por fazer com que seja possível aumentar a produção, diminuindo o custo;
- Flexibilidade: As *wafers* podem ser produzidas em diversos tamanhos, o que acaba por oferecer uma grande flexibilidade na produção, fazendo com que a entidade produtora se consiga adaptar às diversas necessidades;
- Escalabilidade: Com o grande aumento dos dispositivos eletrónicos, a necessidade de semicondutores também aumenta e como as *wafers* permitem a produção de várias unidades ao mesmo tempo, isto faz com que os fabricantes consigam atender à crescente demanda por dispositivos eletrónicos.

Como é óbvio, as *wafers* não são perfeitas e por isso, de seguida, serão apresentadas algumas desvantagens [5]:

- Custo: A produção de *wafers* envolve um alto custo pelo facto de ser necessário um ambiente controlado e também um alto investimento nos equipamentos de produção;

- Fragilidade: As *wafers* são frágeis e ao longo do seu manuseamento existe sempre a possibilidade de as danificar provocando assim prejuízos e/ou atrasos na produção;
- Pureza das *Wafers*: Para se evitar os danos nas *wafers* é necessário manter a alta pureza dos materiais e também controlar o ambiente onde estas são produzidas. Manter a pureza e o ambiente controlado é um processo crítico na sua produção;
- Impacto Ambiental: A produção das *wafers* para além da utilização de vários químicos, que necessitam de ser tratados para evitar a poluição, também são utilizados níveis elevadíssimos de água e energia, o que pode ter um impacto significativo para o meio ambiente.

Na figura seguinte está presente um exemplo de uma *wafer* onde se pode ver a zona ativa da mesma.

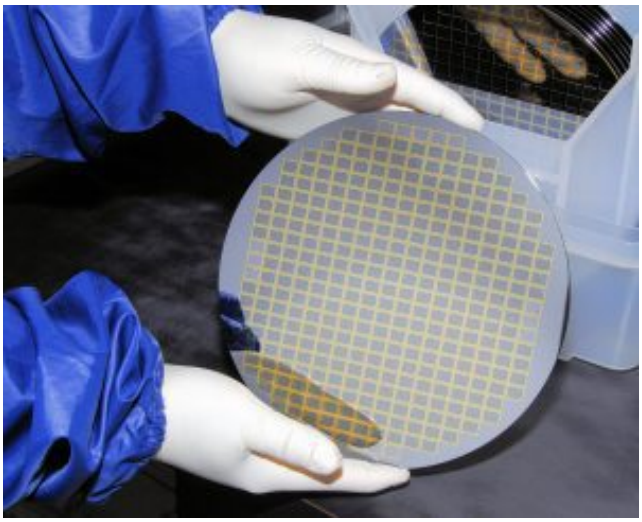


Figura 2.3: Exemplo de uma *wafer* [6].

2.1.2 Produção

Atualmente, a cadeia de um circuito integrado pode ser dividida em cinco etapas: a conceção, o *design*, o *front-end*, o *back-end* e o serviço ao cliente. A fase de conceção, como o nome indica, é onde a ideia inicial é concebida. Geralmente, um *chip* (circuito integrado) é desenvolvido para atender às necessidades do mercado, podendo ser realizado em colaboração com o cliente ou fabricante do produto final. A próxima etapa, conhecida como *design*, envolve o projeto dos circuitos integrados. Durante o *front-end*, o silício é transformado num *chip*. Já a montagem, encapsulamento e teste do circuito integrado são considerados etapas finais que são denominadas de

back-end. Por fim, o serviço ao cliente é fornecido após a conclusão de todas as etapas [7].

De seguida está presente uma figura que demonstra a cadeia de produção de um circuito integrado.

Figura 2.4: Cadeia de um Circuito Integrado [7].

A *Amkor Technology Portugal* (ATEP) é uma organização classificada como *Back-End* embora também tenha alguns processos de *Front-End*. Um dos processos realizados na ATEP é o processo *Embedded Wafer Level Ball Grid Array* (eWLB) que é um processo que surgiu na necessidade de diminuir o tamanho e a altura do *package* conseguindo reduzir os custos. Os produtos produzidos pela *Amkor Technology Portugal* no processo anterior são do tipo *fan-in* e *fan-out* sendo o seu processo dividido em várias etapas:

- Preparação das *wafers* - *Waferprep*;
- *R*econstrução das *wafers* - *Recon*;
- Camadas de redistribuição - RDL;
- Anexação de bolas de solda - *SBA*;
- Empacotamento dos circuitos integrados - *Packing*.

Uma *wafer fan-in* é uma *wafer* onde as ligações elétricas com o exterior apenas existem na área do circuito integrado. Uma *wafer fan-out* é uma *wafer* onde as ligações elétricas com o exterior existem na área do circuito integrado e também na sua periferia. De seguida, está presente na figura a diferença entre *wafers fan-in* e *wafers fan-out*.

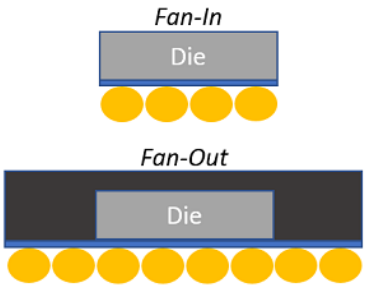


Figura 2.5: *Wafer fan-in vs Wafer fan-out* [8].

Na etapa de preparação das *wafers* (*waferprep*), as *wafers* de silício são recebidas e como o próprio nome sugere, começam-se a preparar para o processo de eWLB. Durante esta etapa, as *wafers* passam por um processo de desbaste da zona não ativa, com o objetivo de reduzir a sua espessura (*grinding*). Em seguida, as *wafers* de produtos do tipo *fan-out* são submetidas a uma operação de singularização dos *dies* (*dicing*) antes de avançarem para a próxima etapa da linha de produção (*recon*). Os *dies* são os pequenos circuitos integrados que compõem uma *wafer*. As *wafers* de produto *fan-in* são direcionadas diretamente para a área de *Redistribution Layer* (RDL). Já na área de *recon*, os *dies* previamente singularizados são colocados num transportador metálico revestido com adesivo por meio das operações de *pick and place*. Antes de passar para a operação de *mold-wafer* a *wafer* passa por um processo de cura para melhorar a adesão dos *dies* na película adesiva. Já no *mold-wafer*, é depositada uma resina líquida chamada de *mold-compound* sobre a *wafer* reconstruída, por meio de um processo de compressão e vácuo, o conjunto adquire a forma de uma *wafer*. Após a cura da resina, a película, colocada anteriormente é removida [7].

Na próxima figura é possível visualizar o processo de *recon*.

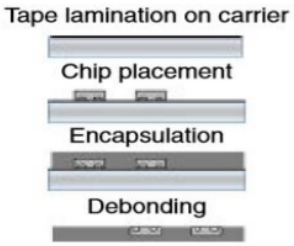


Figura 2.6: Processo de *Recon* [7].

A próxima etapa ocorre na área de RDL, onde as camadas elétricas de cada *die* são redistribuídas por meio de processos físicos e químicos. O processo começa com o aquecimento dos lotes de *wafers* para eliminar qualquer humidade presente. Em seguida, as *wafers* são limpas, na subárea *dry*, utilizando processos de *etching*. O processo de *etching* consiste num processo químico de remoção de impurezas

da superfície da *wafer*. Finalizados os processos de *etching*, os lotes seguem para uma outra subárea, dentro do RDL, denominada de litografia. Nesta subárea, é depositado um material fotossensível sobre a *wafer* para que, posteriormente, esta seja exposta a radiação ultravioleta e seja possível criar pistas elétricas.

Após este processo, o lote segue para uma outra área denominada de *wet*, onde o objetivo é a eletrodeposição do cobre através da adição de uma camada de crescimento. Para finalizar o processo de RDL, é repetida a primeira etapa da subárea litografia, servindo para criar as zonas dos *ball pads*. Concluída a etapa de RDL, os lotes de *wafers* são encaminhados para a área de *Solder Ball Attach* (SBA) para a colocação de bolas de solda em zonas específicas do *die*, utilizando uma máscara apropriada, apelidada de *stencil*. Após o processo de SBA, as *wafers* seguem para a área de *packing*, onde cada *die* é identificado e gravado a laser e por fim colocado no respetivo suporte já individualizado. Cada *die* é inspecionado automaticamente para garantir a conformidade e o produto é embalado para ser expedido para o cliente [7].

2.2 Linguagens de Programação

As linguagens de programação, são instrumentos que permitem aos programadores escrever código de forma mais facilitada para criar um *software*. Cada linguagem de programação possui a sua própria sintaxe, que define a estrutura e a forma correta de escrever instruções. Além disso, elas possuem os seus próprios recursos e bibliotecas que podem ajudar os programadores a realizar tarefas específicas de forma mais simples e eficiente.

Pode-se dizer que existem dois tipos de linguagens de programação sendo a linguagem de programação de baixo nível e linguagem de programação de alto nível. A linguagem de baixo nível é uma linguagem de programação dependente da máquina (*assembly*). O processador executa os programas de baixo nível diretamente, sem a necessidade de um compilador, o que permite que os programas desenvolvidos nessas linguagens sejam executados mais rapidamente [9].

A linguagem de programação de alto nível é projetada para desenvolver programas de *software*, *websites*, entre outros, que sejam *userfriendly* para os programadores. Estes tipos de linguagens de programação requerem tipicamente um compilador para traduzir o programa para a linguagem máquina. A principal vantagem de uma linguagem de alto nível é que ela é fácil de ler, escrever e manter [9] [10].

A escolha de uma linguagem de programação, deve considerar vários fatores, como o tipo de projeto a que se destina, os requisitos técnicos, as preferências pessoais e o ambiente de desenvolvimento disponível, assim como o suporte. Cada linguagem tem as suas vantagens e desvantagens e a escolha certa depende das necessidades e do contexto do projeto em questão.

De seguida, serão então apresentadas algumas linguagens de programação que se adequariam a este tipo de projeto, desenvolvimento de uma interface gráfica.

2.2.1 C#

O C#, Figura 2.7, é uma linguagem de programação recente, orientada a objetos, que foi desenvolvida e impulsionada pela *Microsoft*. Foi lançada em 2000 como parte da plataforma *.NET* e tem sido amplamente utilizada para desenvolver diversos tipos de aplicações, desde aplicações para computadores até aplicações para dispositivos móveis e *web* [11]. O *.NET* é uma plataforma de desenvolvimento de *software* da *Microsoft* que oferece um ambiente de execução, bibliotecas e ferramentas para criar aplicações em várias linguagens de programação [12].

O C# permite aos programadores construir aplicações seguras e robustas que funcionam no ambiente *.NET*. A linguagem tem raízes na família C, o que faz com que programadores familiarizados com linguagens como C, C++, Java e JavaScript se sintam à vontade com essa nova linguagem [11]. Como todas as linguagens, o C# apresenta vantagens e também desvantagens. Posto isto, de seguida serão abordadas algumas vantagens e desvantagens da utilização do C#.

Vantagens[13]:

- Integração com a plataforma *.NET*: O C# é a linguagem principal para o desenvolvimento de aplicações na plataforma *.NET* da *Microsoft*. Isto significa que os programadores podem aproveitar um grande ecossistema do *.NET*, incluindo bibliotecas, *frameworks* e ferramentas.
- Produtividade: O C# foi projetado com foco na produtividade do programador. Ele oferece recursos como por exemplo a gestão automática de memória (*Garbage Collection*);
- Comunidade ativa: O C# possui uma grande comunidade de programadores, o que significa que há muitos recursos disponíveis para auxiliar no processo de desenvolvimento.

Desvantagens[13]:

- Restrição à plataforma *Windows*: O C# tem sido amplamente associado à plataforma *Windows* e ao desenvolvimento de aplicações nesse ambiente. Embora o *.NET Core* tenha expandido o suporte a outras plataformas, como *Linux* e *macOS*, ainda há limitações em comparação com linguagens mais amplamente adotadas, como Java;
- Curva de aprendizagem inicial: O C# possui um conjunto extenso de recursos e uma sintaxe um pouco mais complexa em comparação com algumas outras linguagens de programação;

- Desempenho: Embora o C# tenha um bom desempenho na maioria dos cenários, em algumas situações pode ser menos eficiente do que linguagens de programação como o C++.

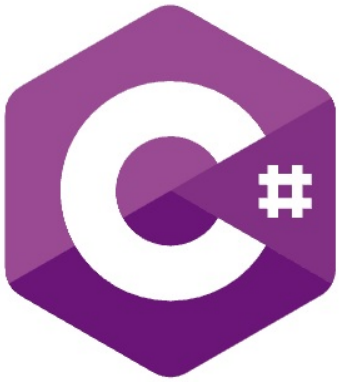


Figura 2.7: C# [9].

2.2.2 Python

Desenvolvida por *Guido van Rossum* em 1991, a linguagem Python, Figura 2.8, tornou-se extremamente popular devido à sua sintaxe simples, tornando-se não só numa escolha ideal para os iniciantes como também para programadores experientes [14]. Uma das principais características do Python é o facto de ser uma linguagem de programação interpretada, isto é, o código fonte é executado diretamente sem a necessidade de ser compilado antes da execução, ou seja, o interpretador lê linha a linha o código fonte e realiza a execução das instruções conforme se encontra programado o código [15]. O Python é usado em várias áreas, como por exemplo o desenvolvimento web, *data science* e inteligência artificial[14]. A seguir, serão abordadas algumas vantagens e desvantagens do uso da linguagem Python.

Vantagens[16]:

- Sintaxe simples: A sintaxe simples e concisa do Python facilita a leitura do código desenvolvido, tornando a linguagem acessível não só para programadores experientes, mas também para iniciantes;

- Amplas bibliotecas e *frameworks*: Uma das grandes vantagens do Python é a existência de uma vasta coleção de bibliotecas e *frameworks*, que facilitam o desenvolvimento de aplicações;

- Comunidade ativa: O Python possui uma das maiores comunidades de programadores, que disponibilizam conteúdos relacionados à linguagem. Há uma

grande quantidade de recursos disponíveis, como documentação, fóruns e tutoriais, o que facilita na hora de programar.

Desvantagens[16]:

- Desempenho relativamente mais baixo: Em situações onde a velocidade de execução é crucial, o Python apresenta algumas limitações quando comparado com outras linguagens.
- Consumo de memória: Quando se trata de tarefas que requerem grande quantidade de memória, o Python não é a escolha mais adequada. Por essa razão, ele não é amplamente utilizado nesse contexto. O consumo de memória do Python também tende a ser elevado, devido à flexibilidade dos seus tipos de dados.

Figura 2.8: Python [9].

2.2.3 Java

O Java, Figura 2.9, foi inventado em 1991 por *James Gosling* da *Sun Microsystems*. É uma linguagem de programação orientada a objetos amplamente utilizada, uma plataforma de *software* que é executada em milhões de dispositivos, incluindo computadores, dispositivos móveis, dispositivos médicos e muitos outros. As regras de sintaxe do Java são baseadas nas linguagens C e C++, o que possibilita aos programadores familiarizados com essas linguagens uma certa facilidade em migrar para Java [17]. O *Java Virtual Machine (JVM)* é uma parte essencial da plataforma Java, sendo responsável por executar os programas Java em diferentes sistemas operativos, oferecendo recursos como bibliotecas para facilitar o desenvolvimento de aplicações[18]. Posto isto, de seguida serão abordadas algumas vantagens e desvantagens da utilização do Java.

Vantagens[19]:

- Portabilidade: O Java é conhecido pela sua capacidade de ser executado em várias plataformas, proporcionando portabilidade às aplicações desenvolvidas. Isto significa que um programa desenvolvido em Java pode ser desenvolvido para um sistema operativo em específico e consegue ser executado em outro sem a necessidade de grandes modificações;
- Segurança: O Java possui recursos integrados de segurança o que permite que os programadores desenvolvam programas seguros, evitando o acesso não autorizado a recursos do sistema protegendo contra ameaças comuns de segurança;
- Comunidade: Como o Java é uma linguagem muito utilizada, esta possui uma grande comunidade de programadores, com uma ampla variedade de bibliotecas, *frameworks* e ferramentas disponíveis facilitando o desenvolvimento de aplicações.

Desvantagens[19]:

- Memória consumida: O Java recorre à utilização do *Java Virtual Machine* o que pode consumir muita memória do sistema e assim, apresentar problemas em sistemas com pouca memória disponível devido à sobrecarga da memória;
- Curva de aprendizagem inicial: O Java apresenta uma sintaxe rica e complexa, o que pode dificultar a sua aprendizagem inicial;
- Sobrecarga de tempo de execução: A execução de código Java envolve uma quantidade de sobrecarga de tempo de execução, o que pode impactar negativamente o desempenho em sistemas com recursos mais limitados.

Figura 2.9: Java [20].

2.2.4 Resumo

A linguagem C# é amplamente utilizada para o desenvolvimento de aplicações *Windows*, jogos e aplicações empresariais, enquanto o Java é comumente usado em aplicações empresariais, desenvolvimento Android e servidores *web*. O Python ganhou popularidade em *data science* e inteligência artificial.

Em termos de sintaxe, o C# e o Java têm uma sintaxe semelhante, enquanto Python se destaca pela sua sintaxe mais simples.

Quanto à portabilidade, o C# historicamente estava mais ligado ao ambiente *Windows*, mas agora é suportado em outras plataformas já o Java, à semelhança do Python, é conhecido por ser altamente portátil.

Em conclusão, a escolha entre uma destas três linguagens de programação depende das necessidades específicas do projeto e preferências pessoais do programador. Cada linguagem possui as suas vantagens e casos de uso distintos, sendo importante considerar esses fatores ao decidir qual linguagem utilizar.

Para este projeto em questão, optou-se por utilizar a linguagem C# devido à sua facilidade de uso, desempenho e também pelo facto de existir uma ampla comunidade de programadores.

2.3 Interface Gráfica do Utilizador

Ao desenvolver uma aplicação, uma das decisões mais importantes é fazer a escolha da linguagem de programação mais adequada para atender às necessidades do projeto.

No entanto, a escolha da linguagem de programação não é a única decisão que deve ser tomada. A interface gráfica também é uma parte essencial de qualquer aplicação. Existem várias *frameworks* que podem ser usadas, cada uma com as suas próprias vantagens e desvantagens. De seguida, serão então apresentadas algumas *frameworks* existentes para a construção de interfaces gráficas (GUIs). No final, ao considerar todas as opções apresentadas, é possível escolher a *framework* mais adequada para criar a interface que atenda às necessidades específicas do projeto.

2.3.1 Windows Forms

O *Windows Forms* é uma biblioteca de classes do *.NET Framework* que permite criar aplicações para computador com interfaces gráficas amigáveis. Com o *Windows Forms*, facilmente se pode criar botões, janelas, menus, caixas de texto, listas, entre outras opções. O *Windows Forms* é uma tecnologia madura e bem estabelecida, que foi introduzida pela *Microsoft* em 2002 com o lançamento do *.NET Framework* 1.0. Desde então, o *Windows Forms* foi amplamente utilizado para criar aplicações em vários setores [21].

Uma das principais vantagens da utilização do *Windows Forms* é a sua facilidade de uso. É possível criar interfaces gráficas usando o *designer* visual, que permite arrastar e soltar componentes, ou escrever o código manualmente. Além disso, o *Windows Forms* é também altamente personalizável, permitindo que sejam criadas interfaces adequadas às necessidades pretendidas. Apesar das suas vantagens o *Windows Forms* também apresenta algumas limitações. Por exemplo, as aplicações desenvolvidas são restritas ao ambiente de um *desktop* e já está um pouco em desuso por se terem criado alternativas tal como o *Windows Presentation Foundation* [22]. Na figura seguinte é possível visualizar um exemplo de uma interface criada com o *Windows Forms*.

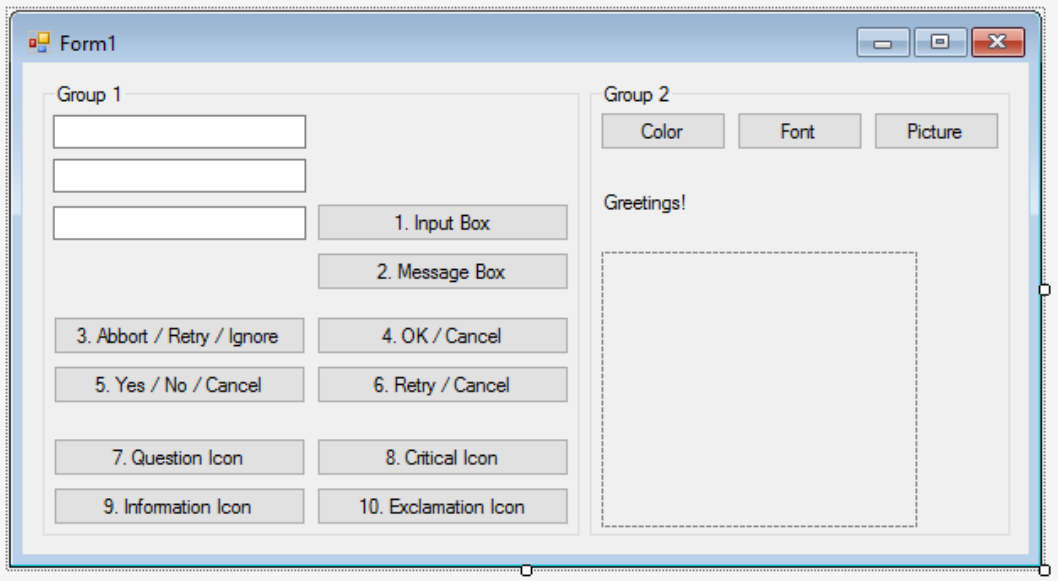


Figura 2.10: Exemplo de uma *GUI* desenvolvida em *Windows Forms* [23].

2.3.2 Windows Presentation Foundation

À semelhança do *Windows Forms*, o *Windows Presentation Foundation (WPF)* é uma *framework* de interface gráfica da *Microsoft* que foi introduzido pela primeira vez em 2006 como parte do *.NET Framework* 3.0 e é utilizado para criar aplicações para o *Windows*. Uma das principais vantagens do *Windows Presentation Foundation* é a utilização da linguagem *Extensible Application Markup Language (XAML)* para definir a interface gráfica. Essa abordagem permite uma clara separação entre a parte gráfica e a lógica da aplicação. Ao utilizar o XAML, os programadores podem descrever a estrutura visual da interface de forma declarativa, o que facilita o *design* e a manutenção do código. Essa divisão entre a parte gráfica e lógica

também permite que *designers* e programadores trabalhem de forma colaborativa, onde os *designers* podem focar no aspeto visual da interface, enquanto os programadores implementam a lógica por trás dela. Essa separação aumenta a eficiência no desenvolvimento e possibilita a criação de interfaces gráficas mais intuitivas e atrativas para os utilizadores. Uma outra vantagem é a sua flexibilidade podendo criar interfaces gráficas complexas, com elementos interativos como animações, gráficos, vídeos ou até mesmo áudio[24].

O WPF conta com um sistema avançado de *data binding* que permite a ligação entre os dados aos seus elementos da interface gráfica. Esta ligação ajuda a criação de aplicações que trabalhem com dados em tempo real [24]. Por outro lado, a maior desvantagem do WPF é que apresenta um desempenho inferior na renderização quando corre em um *hardware* com baixas características gráficas [25].

Em suma, o *Windows Presentation Foundation* permite criar interfaces gráficas ricas em conteúdo e altamente interativas, com uma separação clara entre a lógica de programação e de interface, embora necessite de um *hardware* com algum desempenho. A figura seguinte mostra um exemplo de uma *GUI* desenvolvida em *Windows Presentation Foundation*.

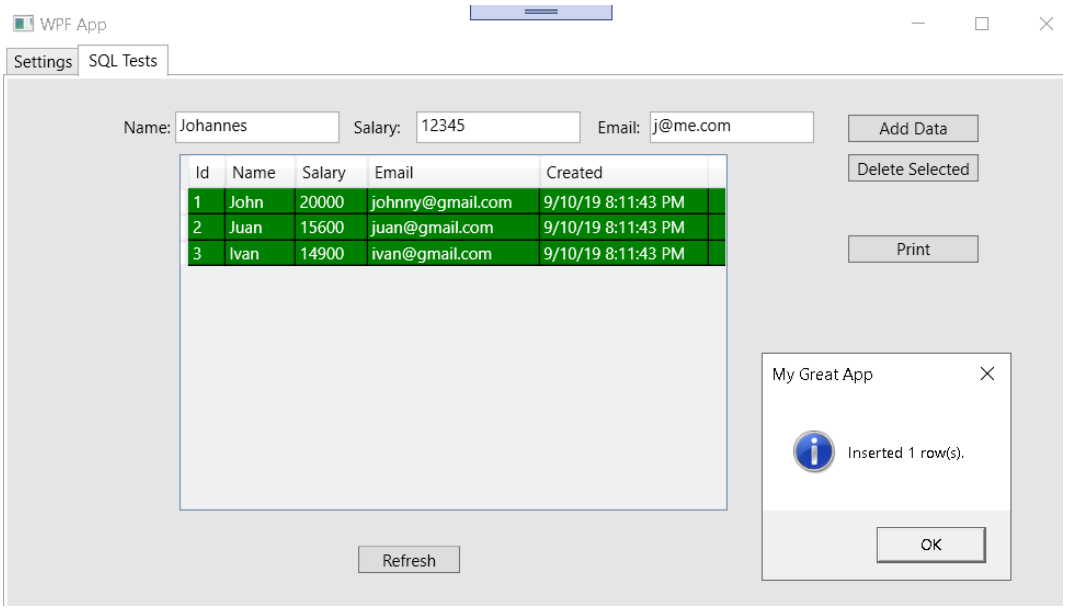


Figura 2.11: Exemplo de uma *GUI* desenvolvida em *WPF* [26].

2.3.3 Qt

O Qt, Figura 2.12, é uma *framework* de desenvolvimento de aplicações multiplataforma para desenvolvimento de interfaces gráficas em C++, criada pela empresa

Trolltech. Uma das características mais notáveis do Qt é a sua portabilidade. O Qt, como foi dito anteriormente, suporta várias plataformas incluindo *Windows*, *masOS*, *Linux*, entre outras. Isso permite que os programadores possam criar aplicações e tenham a possibilidade de as executar em diferentes plataformas sem que necessitem de reescrever o código [27].

Figura 2.12: Qt [27].

O Qt conta com uma longa comunidade de programadores o que faz com que exista muita documentação disponível para facilitar a sua utilização. Por outro lado, quando comparado com outras *frameworks*, o Qt pode apresentar alguma dificuldade no seu desempenho [28]. De seguida está presenta na Figura 2.13 um exemplo de uma *GUI* desenvolvida em Qt.

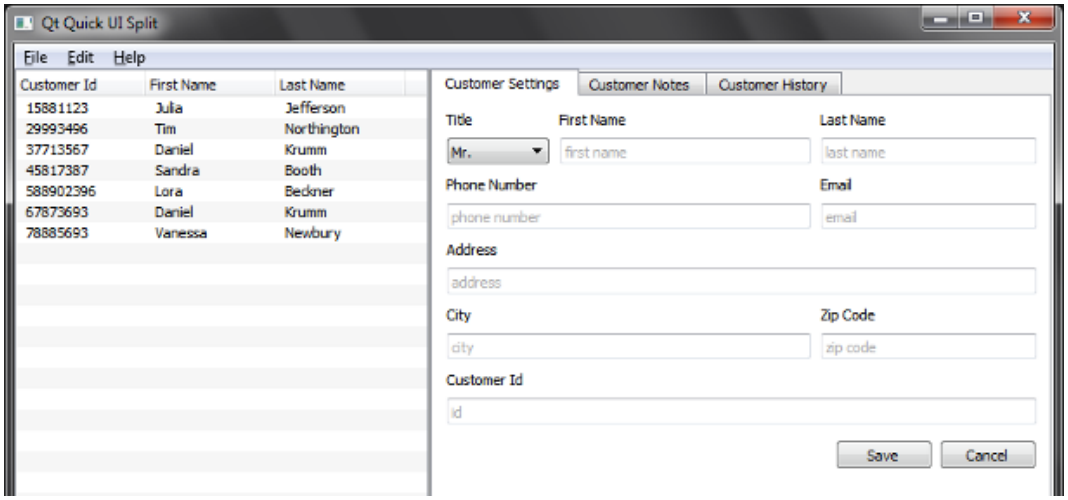


Figura 2.13: Exemplo de uma *GUI* desenvolvida em Qt [29].

2.3.4 GTK+

O *GIMP Toolkit (GTK)*, Figura 2.14, mais conhecido como GTK+, à semelhança das anteriores, é uma biblioteca de interface gráfica multiplataforma. O GTK+ é normalmente usado em ambientes de computadores *Linux* [30].

Uma das principais vantagens do GTK+ é a sua flexibilidade, permitindo que os programadores consigam personalizar não a aparência mas também o comportamento dos *widgets* disponíveis. Uma vantagem quando comparado a outras bibliotecas é que não é necessário pagar taxas de licenças para aplicações comerciais. Por outro lado, ao contrário do Qt, existe uma documentação oficial limitada em alguns casos, o que pode dificultar a solução de problemas que os programadores encontrem [31].

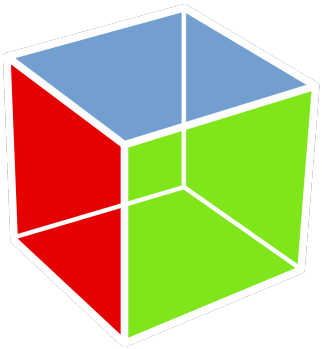


Figura 2.14: GTK+ [27].

Em resumo, o GTK+ é uma biblioteca poderosa e flexível para criar *GUIs* para vários sistemas operativos tendo como maior foco o ambiente *Linux*. De seguida, está presente na figura um exemplo de uma GUI desenvolvida em GTK.

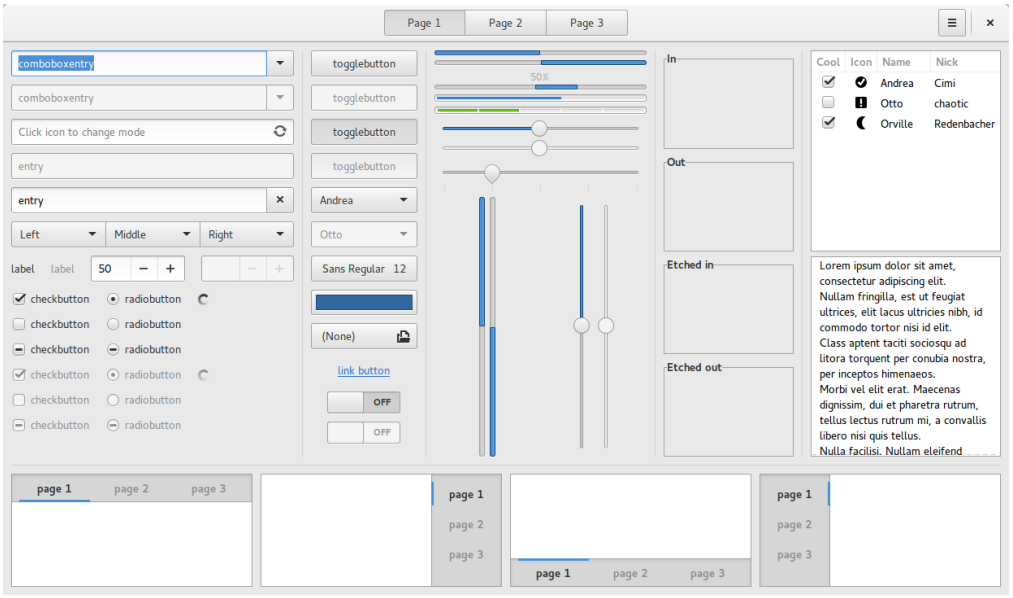


Figura 2.15: Exemplo de uma *GUI* desenvolvida em GTK [32].

2.3.5 Resumo

Relativamente a estas quatro *frameworks*, em termos de facilidade de uso, o *Windows Forms* é considerado o mais acessível de aprender e usar, especialmente para programadores que estão a começar a desenvolver aplicações para *desktop*. O *Windows Presentation Foundation* é mais avançado, quando comparado com o *Windows Forms* e também é relativamente fácil de usar. O Qt e o GTK+ oferecem recursos mais avançados e personalizáveis o que dificulta mais a sua implementação, no contexto em que se enquadra este projeto.

Em termos de desempenho, o *Windows Forms* é um pouco mais limitado em termos de recursos, quando comparado com as outras três *frameworks*, embora ainda seja uma opção viável para aplicações mais simples.

Assim, a escolha de uma destas quatro *frameworks* depende principalmente das necessidades do projeto e do programador. As quatro *frameworks* oferecem recursos únicos e podem ser utilizados para criar diversas aplicações.

Após analisar as diferentes possibilidades, optou-se pela escolha do *Windows Presentation Foundation* devido ao seu equilíbrio entre facilidade de implementação e personalização. Esta *framework* permite criar interfaces gráficas complexas, com elementos interativos, juntamente com o facto de ser relativamente fácil de aprender e usar. Isso torna o *Windows Presentation Foundation* uma opção adequada para o projeto em questão, oferecendo flexibilidade e recursos avançados sem comprometer a usabilidade e eficiência de desenvolvimento.

2.4 Ambientes de Desenvolvimento Integrados

As *Integrated Development Environments (IDEs)*, ou Ambientes de Desenvolvimento Integrados, são ferramentas de *software* projetadas para auxiliar os programadores no processo de criação, edição e compilação de código. Elas oferecem uma interface integrada que reúne várias funcionalidades e recursos essenciais para facilitar o desenvolvimento de aplicações.

2.4.1 SharpDevelop

O *SharpDevelop*, também conhecido como #develop, é um ambiente *open-source* de desenvolvimento integrado leve para o *Windows*. Embora inicialmente tenha sido projetado principalmente para a linguagem de programação C#, neste momento, também oferece suporte a outras linguagens, como por exemplo o *Visual Basic .NET* (VB.NET), C++, F# e Python[33]. Este IDE possui várias vantagens significativas, tais como[34]:

- Compatibilidade com múltiplas linguagens: O *SharpDevelop* permite o desenvolvimento em diversas linguagens da plataforma *.NET*, o que proporciona flexibilidade aos programadores para escolherem a linguagem mais adequada para os seus projetos;
- Interface intuitiva: O *SharpDevelop* possui uma interface de utilizador intuitiva, que facilita a navegação e o acesso às diferentes funcionalidades do IDE. Isso torna a experiência de desenvolvimento mais agradável e produtiva;
- Uso de *plugins*: O *SharpDevelop* suporta o uso de *plugins*, permitindo aos programadores aumentarem as funcionalidades do IDE de acordo com suas necessidades. Os *plugins* adicionais podem adicionar recursos específicos, como suporte a base de dados, ferramentas de análise de código e *frameworks* extra.

Com todas essas vantagens, o *SharpDevelop* torna-se uma opção atraente para programadores que procuram um IDE gratuito, *open-source* e repleto de recursos para o desenvolvimento em linguagens *.NET*.

Figura 2.16: SharpDevelop [35].

2.4.2 Visual Studio

O *Visual Studio* é um ambiente de desenvolvimento integrado altamente conceituado, desenvolvido pela *Microsoft*. Este IDE oferece um grande número de recursos, linguagens de programação e funcionalidades que tornam o processo de desenvolvimento de *software* mais produtivo e eficiente [36].

De seguida estão presentes três vantagens do uso deste ambiente de desenvolvimento [36]:

- Extensibilidade: A extensibilidade do *Visual Studio* é excecional, permitindo que os programadores personalizem e adaptem o IDE de acordo com suas necessidades por meio de extensões e *plugins*. Essa extensibilidade não é apenas resultado do suporte oficial da *Microsoft*, mas também é impulsionada por uma comunidade ativa. Essa combinação de contribuições resulta num ecossistema sólido para o *Visual Studio*;
- Ampla compatibilidade e suporte multiplataforma: O *Visual Studio* suporta várias linguagens de programação, como C#, C++, Python e muitas outras. Além disso, ele oferece suporte em diversas plataformas incluindo o *Windows*, *macOS* e *Linux* o que o torna uma escolha muito popular para programadores em várias plataformas;
- Recursos poderosos: O *Visual Studio* disponibiliza uma vasta gama de funcionalidades poderosas como por exemplo a oferta de sugestões de código inteligente e um *debugging* capaz de identificar e corrigir problemas no código de forma eficiente.

Com todas essas vantagens, o *Visual Studio* torna-se uma opção atraente para programadores que procurem um editor de código repleto de recursos para o desenvolvimento em várias linguagens de programação.

Figura 2.17: *Visual Studio* [37].

2.4.3 Resumo

O *Visual Studio* e o *SharpDevelop* são dois ambientes de desenvolvimento integrados amplamente utilizados, mas possuem diferenças significativas. A escolha entre o *Visual Studio* e o *SharpDevelop* dependerá das necessidades específicas de desenvolvimento, das linguagens de programação utilizadas e das preferências individuais. O *Visual Studio* é altamente personalizável, compatível com várias linguagens e adequado para diversos cenários de desenvolvimento. Por outro lado, o *SharpDevelop* é mais focado para o desenvolvimento em linguagens *.NET* e oferece recursos específicos para essa plataforma. Existem várias opções de linguagens e *frameworks* para abordar o problema proposto. O primeiro passo será escolher uma linguagem de programação de alto nível, isto é, que tenha capacidade de lidar com um grande volume de dados. Por este pressuposto e por se tratar de uma linguagem de alto nível capaz para o desenvolvimento de aplicações interativas, a escolha recai sobre a linguagem C#, como referido em cima. De seguida, é necessário escolher uma plataforma que permita criar a interface gráfica na linguagem escolhida e onde seja possível apresentar graficamente os dados paramétricos . Após uma análise das opções, optou-se pela *Windows Presentation Foundation* pois foi criada pela *Microsoft* e incorpora a plataforma *.NET* tal como o C#. Por fim, optou-se por utilizar como ambiente de desenvolvimento o *SharpDevelop* para desenvolver a aplicação devido a este ser um ambiente de desenvolvimento integrado especializado em linguagens *.NET*.

Capítulo 3

Parte Prática

Neste capítulo, será abordada a parte prática do desenvolvimento do projeto. Inicialmente, será abordada a medição estatística do percentil e o propósito subjacente à sua utilização na visualização dos dados a apresentar. Em seguida, será apresentada a forma como a interface gráfica foi estudada e criada, com o objetivo de minimizar reformulações futuras. Posteriormente, será realizada uma abordagem individual aos diferentes componentes que compõem a interface idealizada para o utilizador, mostrando desde o seu design até pequenos excertos de código desenvolvido. Na secção seguinte, será abordada a gestão de dados em memória realizada pela aplicação, com o intuito de otimizar o seu uso. Por fim, serão abordadas as estratégias implementadas para tornar a aplicação mais robusta, prevenindo erros e falhas no sistema. Na figura seguinte está presente a visão geral da aplicação.

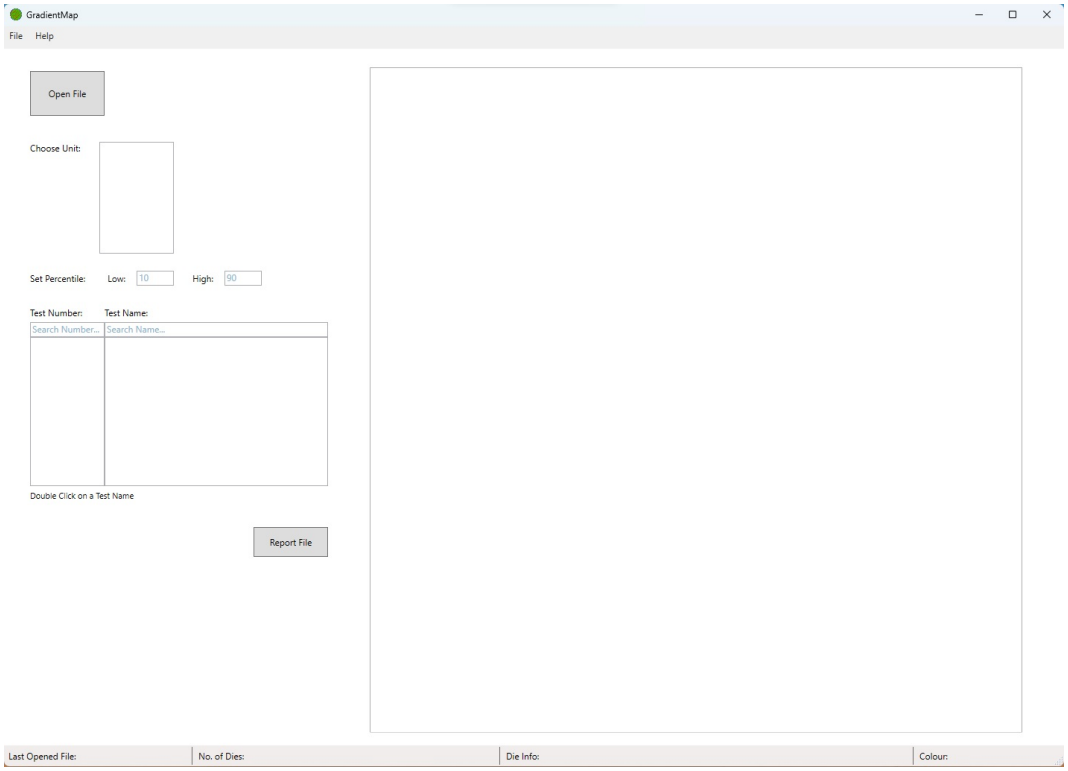


Figura 3.1: Visão geral da aplicação

3.1 Percentil

Em determinados testes, o intervalo de valores no qual os resultados devem estar eram muito amplos e os valores medidos por vezes eram muito próximos uns dos outros o que dificultava a identificação de padrões no desenho do *wafermap*. Para ultrapassar esta limitação, implementou-se o uso de percentis que fornecem uma perspetiva mais relevante dos dados a analisar. A figura seguinte mostra onde o utilizador pode alterar os valores dos diferentes percentis a calcular.

Figura 3.2: Design do Percentil

Um percentil é uma medida estatística que divide uma distribuição de dados em partes iguais ou proporcionais. É uma forma de entender a posição relativa de um valor dentro de um conjunto de dados.

Por exemplo, o percentil 25 indica o valor abaixo do qual 25% dos dados estão localizados, enquanto o percentil 75 indica o valor abaixo do qual 75% dos dados estão localizados. Estes percentis dividem a distribuição fornecendo informações sobre a posição relativa dos valores dentro do conjunto total de dados. Além disso,

estes percentis são úteis para entender a dispersão dos dados, identificar a variabilidade e verificar se existem valores extremos ou discrepantes do resto dos valores melhorando, assim, o gradiente de cores no desenho do *wafermap*.

O *wafermap* é composto por diversos quadrados com um gradiente de cores sendo ele composto por três cores principais: verde, amarelo e laranja. O verde e o laranja representam os extremos dos percentis, enquanto o amarelo representa os valores centrais dentro do intervalo estabelecido. Os quadrados coloridos a azul são aqueles que excedem os limites mínimos e máximos dos testes, tornando-se assim falhas. Os quadrados em branco representam quadrados que já não fazem parte da *wafer*, pois foram previamente removidos devido a falhas apresentadas.

Para calcular o percentil, é necessário seguir alguns passos. Primeiramente, os dados devem ser ordenados em ordem crescente, para que seja possível determinar a posição do percentil desejado.

A fórmula do percentil começa calculando a posição (P) do percentil desejado, utilizando a seguinte fórmula:

$$P = \frac{n \cdot \text{percentil}}{100}$$

Na fórmula, "n"representa o número total de observações no conjunto de dados e "percentil"é o valor do percentil desejado. Uma vez que a posição do percentil tenha sido calculada, é importante verificar se ela é um número inteiro. Se for, o valor do percentil corresponde ao valor na posição P no conjunto de dados ordenado. Caso a posição não seja um número inteiro, o valor do percentil será a média dos valores nas posições P e P+1.

No excerto de código seguinte está presente o método que trata de fazer o cálculo dos percentis dos valores. Para o método calcular o percentil apenas necessita de dois argumentos, sendo o primeiro a lista de valores organizados de forma crescente e segundo o valor do percentil desejado.

```
1 public static double GetPercentil(this List<double> list, double
      percent)
2 {
3     int n = list.Count;
4     double i = (n - 1) * percent + 1;
5     return i.CompareTo(1d) == 0 ? list[0] : (i.CompareTo(n) == 0 ?
       list[n - 1] : list[(int)i - 1] + (i - (int)i) * (list[(int)i]
      - list[(int)i - 1]));
6 }
```

Listagem 3.1: Função do percentil

3.2 Interface Gráfica

Para que o utilizador pudesse interagir com a aplicação desenvolvida, foi desenhada e implementada uma interface. A interface gráfica desta aplicação foi desenvolvida com recurso ao *Windows Presentation Foundation* utilizando o recurso do *grid*, o que permitiu que a aplicação se adaptasse de forma adequada tanto quando maximizada como quando aberta em outros monitores de menores resoluções, evitando qualquer tipo de desconfiguração visual. O *grid* é uma estrutura bidimensional composta por linhas e colunas, onde os elementos da interface são posicionados em células específicas. Ele oferece a capacidade de definir o número e a largura das colunas, bem como o número e a altura das linhas.

Uma das principais vantagens do *grid* é a sua capacidade de se adaptar a diferentes tamanhos e resoluções de ecrãs. Ao redimensionar a janela ou utilizar monitores com resoluções diferentes, o *grid* ajusta automaticamente o tamanho das células e o posicionamento dos elementos, garantindo uma interface adaptativa. No excerto de código seguinte, é possível observar a divisão da janela em três partes, sendo a primeira e a última para as barras, superior e inferior, introduzidas e a segunda para os restantes elementos da interface.

```
1     <Grid.RowDefinitions>
2       <RowDefinition Height="Auto" /> <!-- Barra superior -->
3       <RowDefinition Height="*" />  <!-- Conteudo principal -->
4       <RowDefinition Height="Auto" /> <!-- Barra inferior -->
5     </Grid.RowDefinitions>
```

Listagem 3.2: Código do *Grid*

Através da análise do código anterior é possível perceber que ambas as barras, ao contrário da linha referente ao conteúdo principal, foram definidas com "Auto". Quando uma linha tem a altura definida como "Auto", ela vai-se ajustar automaticamente para acomodar o tamanho do seu conteúdo. Por outro lado, quando se utiliza o asterisco, "*", significa que a linha vai preencher todo o espaço disponível naquele *grid* em específico. Isto significa que a altura da linha será distribuída de forma proporcional em relação às duas outras linhas. Após definir as três linhas com diferentes alturas, é possível criar subcolunas e sublinhas dentro de cada uma delas para organizar melhor o aspeto da interface. No excerto de código seguinte, é possível observar a criação de duas colunas, dentro da primeira coluna, ou seja, de forma hierárquica, onde foram adicionadas mais oito linhas, sendo cada uma para um elemento diferente da interface.

```
<!-- Conteudo Principal -->
<Grid Grid.Row="1">
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>


  <!-- Grid 1 -->
  <Grid Grid.Column="0" Margin="10">
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
```

Listagem 3.3: Código do *Grid*

3.3 Elementos da Interface

Os elementos da interface têm um papel fundamental numa aplicação. Os botões, as *labels*, as caixas de pesquisa e outros componentes tem uma grande importância na usabilidade e na experiência do utilizador. Eles facilitam a interação e navegação do utilizador pela aplicação, permitindo que realizem ações ou insiram informações importantes para o funcionamento da aplicação. É também através dos vários elementos de interação que é possível criar uma experiência mais fácil e intuitiva para os utilizadores aumentando assim a sua satisfação ao utilizar a aplicação.

3.3.1 Botões

Os botões são essenciais nas interfaces das aplicações, pois permitem que os utilizadores interajam e realizem ações específicas de forma fácil e intuitiva. Eles desempenham um papel importante na usabilidade, fornecendo *feedback* visual e em algumas circunstâncias tátil ao usar os botões do rato.

3.3.1.1 Botão Open File

Numa primeira fase para se conseguir selecionar um ficheiro de dados, foi necessário criar uma forma de se escolher o ficheiro pretendido. Foi então que, neste contexto,

se criou um botão com a funcionalidade de seleção de um ficheiro. A figura seguinte, mostra o botão criado na interface para dar início ao processo de seleção e leitura do ficheiro.

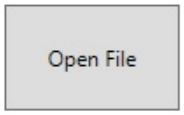


Figura 3.3: Design do botão Open File

Para possibilitar a seleção do ficheiro desejado, o utilizador deve premir o botão adequado, o que abrirá uma janela *pop-up* para escolher o ficheiro pretendido. Segue-se abaixo o excerto de código correspondente à ação descrita anteriormente:

```
1 OpenFileDialog openFileDialog1 = new OpenFileDialog();
2
3 openFileDialog1.Filter = "Access files (*.csv.zip)|*.csv.zip";
4 openFileDialog1.RestoreDirectory = true;
5 openFileDialog1.Multiselect = false;
6
7 if (openFileDialog1.ShowDialog() == true)
```

Listagem 3.4: Código da escolha do ficheiro

Para a escolha do ficheiro apenas será aceite a seleção de um ficheiro e terá de ter a extensão ".csv.zip". Após a seleção do ficheiro escolhido pelo utilizador, segue-se então a abertura do mesmo, como se pode ver no excerto de código seguinte.

```
1 var zip = new ZipInputStream(File.OpenRead(@_tdfile));
2 var filestream = new FileStream(@_tdfile, FileMode.Open,
      FileAccess.Read, FileShare.Read, 4096);
3 ZipFile zipfile = new ZipFile(filestream);
4 ZipEntry item;
5 while ((item = zip.GetNextEntry()) != null)
6 {
7   using (StreamReader s = new StreamReader(zipfile.GetInputStream(
      item)))
8   {
```

Listagem 3.5: Código de leitura do ficheiro

Neste excerto de código, é realizada a leitura de um ficheiro ZIP. Inicialmente, é criada uma instância do objeto responsável pela leitura de ficheiros ZIP. Esse objeto recebe o caminho do ficheiro como parâmetro. A partir daí, é utilizado um *loop* para percorrer cada item dentro do ficheiro ZIP. A cada iteração, o conteúdo do item é lido e processado. O *loop* é finalizado quando não houver mais itens dentro do ficheiro para ler.

Em resumo, o excerto de código do botão *Open File* abre um ficheiro ZIP que contém um ficheiro Excel e lê, linha a linha, o conteúdo desse mesmo ficheiro. Isto permite que seja feito o tratamento dos dados armazenados no ficheiro Excel.

3.3.1.2 Botão Report File

Com o intuito de criar relatórios das *wafers* estudadas, implementou-se um botão que permite ao utilizador gerar um documento PDF contendo as informações importantes sobre a *wafer*. Esta funcionalidade oferece uma forma conveniente e organizada de reunir os dados relevantes num único documento para consulta futura. A figura seguinte mostra o design do botão *Report File*.

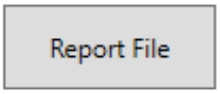


Figura 3.4: Design do botão *Report File*

Ao clicar com o botão do lado esquerdo do rato no botão *Report File*, o *software* reúne automaticamente as informações necessárias para criar o relatório. Estas informações incluem o nome do ficheiro, o nome da *wafer*, o nome do teste escolhido para análise, os dois percentis utilizados e também o *wafermap* correspondente. De seguida, é apresentado o excerto de código desenvolvido para que seja possível criar o ficheiro PDF quando o botão é acionado.

```
void pdf_Click(object sender, RoutedEventArgs e)
{
    try
  {
        PdfDocument document = new PdfDocument();
        PdfPage page = document.AddPage();
    XGraphics gfx = XGraphics.FromPdfPage(page);
        XFont font = new XFont("Arial", 10, XFontStyle.Regular);
    XFont title = new XFont("Arial", 15, XFontStyle.Bold);
        XRect textRect = new XRect(50, 50, page.Width - 100, page.
     Height - 100);

        //Text
        //Image

        document.Save(caminhoFicheiro);
        MessageBox.Show("Report created successfully");
    System.Diagnostics.Process.Start(caminhoFicheiro);
    }
    catch (Exception ex)
  {
    MessageBox.Show("Error creating the report\n    (Close open
     PDFs)");
  }
  finally
  {
    if (document != null)
    {
      document.Close();
    }
  }
 }
```

Listagem 3.6: Código do botão *Report File*

Ao ser acionado o evento do botão, o código cria um documento em formato PDF. É adicionada uma página vazia a esse documento e, em seguida, é inicializado um objeto gráfico para permitir desenhar no documento. São definidas duas fontes necessárias, uma para o título e outra para as informações relativas à *wafer*. Após adicionar o conteúdo desejado, o documento é guardado no ambiente de trabalho do computador. Por fim, caso o documento tenha sido criado com sucesso, o relatório é aberto.

3.3.2 Etiquetas

As etiquetas, *labels*, desempenham um papel fundamental numa interface, pois, embora sejam meramente informativas, como o nome indica, são elas que vão oferecer informações sobre os restantes elementos interativos, como caixas de pesquisa, listas, entre outros. O uso de *labels* foi várias vezes implementado na interface em questão para, de certa forma, tornar a experiência do utilizador mais simples e objetiva. Na figura seguinte é possível visualizar, sublinhado a vermelho, algumas *labels* presentes na interface.

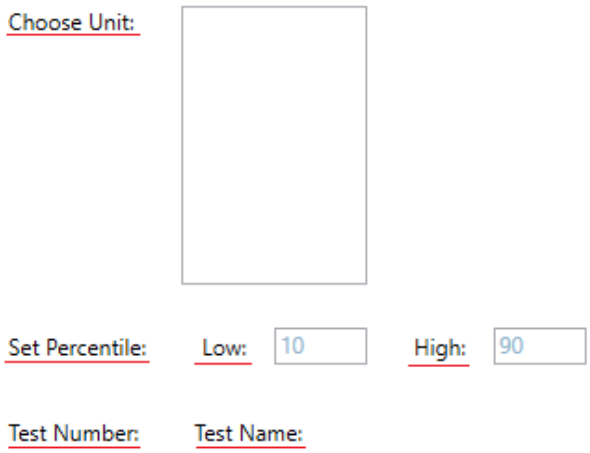


Figura 3.5: *Labels* presentes na interface

- *Choose Unit* (Escolher Unidade): Esta *label* indica ao utilizador que a seleção das unidades deve ser feita na lista disponível à sua direita;
- *Set Percentile* (Definir Percentil): Esta etiqueta informa ao utilizador que é possível definir o valor do percentil nos campos disponíveis;
- *High* (Alto): Refere-se ao valor alto do percentil, ou seja, o limite superior do intervalo estatístico a ser definido;
- *Low* (Baixo): Refere-se ao valor baixo do percentil, ou seja, o limite inferior do intervalo estatístico a ser definido;
- *Test Number* (Número do Teste): Esta etiqueta indica ao utilizador que o conteúdo presente na barra de pesquisa e na lista abaixo dela diz respeito aos números dos testes;
- *Test Name* (Nome do Teste): Esta etiqueta indica ao utilizador que o conteúdo presente na barra de pesquisa e na lista abaixo dela diz respeito aos nomes dos testes.

3.3.3 Caixas de Texto

As caixas de texto, à semelhança dos outros elementos, desempenham um papel importante nas interfaces de utilizador, permitindo não só inserir como também visualizar informações. Por norma, são áreas retangulares interativas onde se pode introduzir texto ou ver dados relevantes. Podem ser usadas em formulários, campos de pesquisa e mensagens. Além disso, as caixas de texto podem exibir informações importantes, como resultados de pesquisa ou até mesmo uma mensagem de erro. Podem ter recursos adicionais, como a validação do conteúdo de entrada e sugestões de texto.

3.3.3.1 Caixa de Entrada de Texto

Foram implementadas duas caixas de texto que permitem ao utilizador definir os valores mínimo e máximo do cálculo do percentil, abordado no início do capítulo, para que consiga obter melhores resultados no momento de imprimir o *wafermap*. O *design* das duas caixas de texto está presente na figura seguinte.

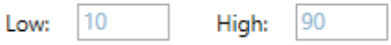


Figura 3.6: *Design* das caixas de texto do percentil

A primeira caixa de texto é destinada à introdução do valor mínimo do percentil. Aqui, o utilizador pode inserir o valor numérico correspondente ao percentil mínimo desejado. Caso não seja inserido nenhum valor, a caixa de texto exibirá uma marca d'água com o valor padrão definido de 10%, que será utilizado como o valor *default* no cálculo do percentil mínimo. A seguir apresenta-se o excerto de código que faz a leitura do conteúdo da caixa de texto e calcula o percentil baseado nesse valor.

```
1  LowPercentil.Text = new string(LowPercentil.Text.Where(char.
       IsDigit).ToArray());
2  LowPercentil.SelectionStart = LowPercentil.Text.Length;
3  if(LowPercentil.Text != "")
4  {
5       double _lowpercentil = Convert.ToDouble(LowPercentil.Text);
6
7    if(_lowpercentil > 100)
8    {
9      MessageBox.Show("Low value must be higher than 0 and lower
       than 100. \nDefault value has been assigned (10)");
10     _lowpercentil = 0;
11     _percentilInfo.setlowpercentil(_lowpercentil);
12   }
13   else
14   {
15     _percentilInfo.setlowpercentil(_lowpercentil);
16   }
17 }
18 else
19 {
20   double _lowpercentil = 0;
21   _percentilInfo.setlowpercentil(_lowpercentil);
22 }
```

Listagem 3.7: Código da caixa de texto do *Low Percentil*

A segunda caixa de texto tem o mesmo objetivo da primeira, com a diferença de que, em vez de se definir o valor mínimo, agora é possível definir o valor máximo pretendido. Mais uma vez, tal como a primeira caixa de texto, a segunda caixa também possui uma marca d'água com o valor padrão definido, que neste caso é de 90%. Isto permite calcular o percentil máximo quando nenhum valor é introduzido. O excerto de código apresentado a seguir é referente ao cálculo do valor máximo do percentil.

```
1  HighPercentil.Text = new string(HighPercentil.Text.Where(char.
       IsDigit).ToArray());
2  HighPercentil.SelectionStart = HighPercentil.Text.Length;
3  if(HighPercentil.Text != "")
4  {
5    double _highpercentil = Convert.ToDouble(HighPercentil.Text);
6    if(_highpercentil > 100)
7    {
8      MessageBox.Show("High value must be higher than 0 and lower
       than 100. \nDefault value has been assigned (90)");
9      _highpercentil = 0;
10     _percentilInfo.sethighpercentil(_highpercentil);
11   }
12   else
13   {
14     _percentilInfo.sethighpercentil(_highpercentil);
15   }
16 }
17 else
18 {
19   double _highpercentil = 0;
20   _percentilInfo.sethighpercentil(_highpercentil);
21 }
```

Listagem 3.8: Código da caixa de texto do *High Percentil*

Em resumo, as duas caixas de texto presentes na interface desempenham um papel fundamental na definição dos valores mínimo e máximo do percentil. O utilizador tem a liberdade de escolher quais são os valores do mínimo e máximo que mais lhe convém para o teste que pretende analisar. Por outro lado, se o utilizador não pretender introduzir nenhum valor, o cálculo será feito com os valores padrão predefinidos.

3.3.3.2 Caixa de Entrada de Texto

Reconhecendo que a quantidade de testes pode ser significativa, foram implementados recursos adicionais para facilitar a seleção dos testes desejados. Para isso, foram incorporadas duas caixas de pesquisa (*search box*), uma para a lista de números e outra para a lista de nomes de testes. De seguida está presente uma figura que mostra as duas *search box* implementadas.

Figura 3.7: Design das *Search Box*

As caixas de pesquisa permitem que o utilizador filtre rapidamente os testes com base em critérios específicos. Por exemplo, se o utilizador conhece parte do número ou do nome do teste, ele pode digitar essas informações na caixa de pesquisa correspondente e os resultados serão atualizados em tempo real, exibindo apenas os testes que correspondem aos critérios de pesquisa fornecidos. Isso simplifica a localização do teste desejado em casos de grandes conjuntos de dados ou nomes complexos.

Estas funcionalidades foram implementadas com o objetivo de otimizar a experiência do utilizador e garantir que ele dispõe das ferramentas necessárias para encontrar e selecionar os testes de interesse, mesmo em cenários com um grande número de opções disponíveis. O código que está associado à *search box* dos números é apresentado a seguir.

```
1  private void searchNumberTextBox_TextChanged(object sender,
       TextChangedEventArgs e)
2  {
3      searchBox.Text = new string(searchBox.Text.Where(char.IsDigit)
       .ToArray());
4      if(_emptyBox == true)
5      {
6          string searchText = searchBox.Text;
7          NumberList.Items.Filter = item => (item as string).
       Contains(searchText);
8          if(searchText == "")
9          {
10             TextFilter.Text = "*";
11             mtestheader.UpdateTestList(SoftBin.Text, TextFilter.
       Text, NumberFilterTest.Text, munitlist.enabledunits());
12             NameList.ItemsSource = mtestheader.GetTestList();
13         }
14         foreach(string items in NumberList.Items)
15         {
16             if(items == searchBox.Text)
17             {
18                 mtestheader.UpdateTestList(SoftBin.Text,
       TextFilter.Text, searchBox.Text, munitlist.enabledunits());
19                 NameList.ItemsSource = mtestheader.GetTestList();
20             }
21         }
22     }
23 }
```

Listagem 3.9: Código da caixa de pesquisa dos números dos testes

Este código é acionado quando o conteúdo da caixa de pesquisa é alterado. Inicialmente, é garantido que apenas são aceites números por se tratar de uma *search box* referente aos identificadores numéricos dos testes.

Caso o utilizador não altere o conteúdo da caixa de pesquisa, as listas dos números e dos nomes dos testes exibirão todo o seu conteúdo. No entanto, se houver algum conteúdo na caixa de pesquisa, a lista apresentada em baixo é atualizada dinamicamente para exibir apenas os números dos testes que contem a informação correspondente à inserida na caixa de pesquisa.

Por fim, se houver um único número na lista que corresponda exatamente ao valor inserido na caixa de pesquisa, apenas esse número e o seu nome correspondente serão exibidos nas listas. De seguida, está presente o excerto de código referente à outra caixa de pesquisa, sendo ela, relacionada com a lista de nomes de teste.

```
1  private void searchTextBox_TextChanged(object sender,
       TextChangedEventArgs e)
2  {
3      if(_emptyBox == true)
4      {
5          string searchText = searchTextBox.Text;
6          NameList.Items.Filter = item => (item as string).Contains(
       searchText);
7          if(searchText == "")
8          {
9              TextFilter.Text = "*";
10             mtestheader.UpdateTestList(SoftBin.Text, TextFilter.
       Text, NumberFilterTest.Text, munitlist.enabledunits());
11             NumberList.ItemsSource = mtestheader.GetTestNumberList
       ();
12         }
13         foreach(string items in NameList.Items)
14         {
15             if(items == searchTextBox.Text)
16             {
17                 mtestheader.UpdateTestList(SoftBin.Text,
       searchTextBox.Text, NumberFilterTest.Text, munitlist.
       enabledunits());
18                 NumberList.ItemsSource = mtestheader.
       GetTestNumberList();
19             }
20         }
21     }
22 }
```

Listagem 3.10: Código da caixa de pesquisa dos nomes dos testes

À semelhança da caixa de pesquisa dos números dos testes, a caixa de pesquisa dos nomes funciona de forma semelhante. A principal diferença é que a caixa de pesquisa dos nomes aceita caracteres alfanuméricos, pois a sua função é filtrar os nomes dos testes.

Podemos observar que ambas as caixas de pesquisa têm o mesmo princípio de funcionamento, permitindo ao utilizador inserir um texto para realizar a pesquisa e filtrar os resultados correspondentes.

3.3.4 Listas

As listas são fundamentais nas interfaces de utilizador. Elas são capazes de oferecer uma forma eficiente e compacta de apresentar opções de seleção numa listagem vertical. As listas facilitam a seleção, poupam espaço, permitem uma fácil leitura e possibilitam a ordenação e filtragem dos itens apresentados.

3.3.4.1 Listas dos Números e Nomes dos Testes

Foram utilizadas duas *listbox* para a apresentação dos nomes dos testes realizados. A primeira *listbox* é dedicada à exibição dos números correspondentes a cada teste, enquanto a segunda *listbox* é responsável por apresentar os nomes associados a esses testes.

Esta abordagem da utilização de duas *listbox*, permitiu uma organização mais clara e intuitiva dos dados, tornando mais fácil para os utilizadores identificarem, em conjunto com as barras de pesquisa, os números e os seus correspondentes nomes. Isto porque muitas vezes se conhece o número do teste a analisar, mas não se conhece o nome ou vice-versa. Estas listas possuem a capacidade de se atualizarem dinamicamente à medida que o utilizador interage com a restante interface. Por exemplo, caso o utilizador altere as unidades pretendidas, as *listbox* vão se atualizar em tempo real proporcionando uma melhor experiência do utilizador.

A próxima figura mostra as duas *listbox* com o conteúdo correspondente de cada uma.

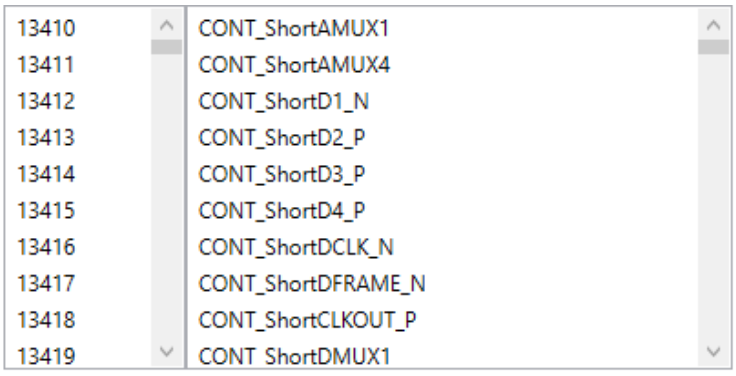


Figura 3.8: *ListBox* Número e Nome dos Testes

A combinação destas duas *listbox* e das duas caixas de pesquisa anteriormente referidas oferece, ao utilizador, flexibilidade e agilidade na seleção do teste desejado. Ele pode optar por navegar diretamente pelas listas de números ou nomes de teste, caso esteja familiarizado com eles, ou utilizar as caixas de pesquisa para facilitar a busca com base em critérios específicos.

3.3.4.2 Lista de Parâmetros

Nesta aplicação, é necessário processar e analisar os dados obtidos a partir do ficheiro Excel anteriormente carregado. No entanto, nem todos os testes presentes no ficheiro são relevantes para os objetivos específicos desta aplicação. Apenas os testes que possuem unidades elétricas de medida no SI é que são de interesse. A Figura 3.9 mostra a *listbox* onde são apresentadas as unidades.

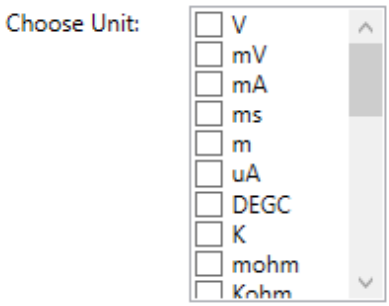


Figura 3.9: *ListBox* de Parâmetros

Para facilitar a seleção do teste desejado, foi implementado uma *listbox* interativa na interface da aplicação. A *listbox* apresenta uma lista de opções de parâmetros (associados à respetiva unidade) presentes no ficheiro Excel, permitindo ao utilizador selecionar uma ou mais opções a serem consideradas. Ao selecionar os parâmetros desejados na *listbox*, a aplicação é capaz de filtrar e mostrar apenas os testes que possuem essas unidades associadas. Isto permite que a análise seja direcionada para os testes relevantes, evitando o excesso de informação na interface.

O excerto de código seguinte mostra um *loop for* que percorre os parâmetros contidos em *munitlist* e, caso o valor seja diferente de *null*, esse parâmetro é adicionado à *listbox*. Isto permite com que exista uma filtragem, como referido anteriormente.

```
1 checkedListBox.Items.Clear();
2 for(int i=0; i<munitlist.getunitcount(); i++){
3     if(munitlist.getunitvalue(i).unit != null)
4         checkedListBox.Items.Add(munitlist.getunitvalue(i).unit);
5 }
```

Listagem 3.11: Código de adição de unidades à *ListBox*

3.3.5 Tab Control

Com o objetivo de fornecer aos utilizadores uma visualização clara e flexível dos *wafermaps*, implementou-se a funcionalidade *TabControl*. O *TabControl* é uma funcionalidade do *Windows Presentation Foundation* poderosa que permite organizar visualmente as informações em guias separadas, tornando mais fácil para o utilizador alternar entre diferentes mapas apenas com um clique. Esta ferramenta permite que os utilizadores consigam ter vários separadores abertos simultaneamente, cada um com um *wafermap* diferente.

Esta abordagem tem várias vantagens. Em primeiro lugar, permite ao utilizador comparar diferentes *wafermaps* lado a lado, facilitando a identificação de padrões ou discrepâncias entre eles. Por exemplo, se existirem várias *wafers* de uma mesma produção, o utilizador consegue abrir vários separadores correspondentes a diferentes *wafers* e avaliar comparativamente o mesmo teste. Além disso, a possibilidade de ter vários separadores permite ao utilizador avaliar os padrões de diferentes *wafers* e de diferentes testes sem ter de fechar e voltar a abrir o mapa. De seguida, é apresentado o excerto de código correspondente ao início do método que faz o desenho do *wafermap* aplicando o gradiente de cores.

```
1  public void Refresh()
2  {
3      _CurrentNx = _Nx;
4      _CurrentNy = _Ny;
5
6      width = 850;
7      height = 850;
8
9      X0 = (int)((double)width / (double)1000 * _WaferEdgeExclusion)
       ;
10     Y0 = (int)((double)height / (double)1000 * _WaferEdgeExclusion
       );
11     Xinc = ((double)width - (double)(X0 * 2) - 1d) / (double)
       _CurrentNx;
12     Yinc = ((double)height - (double)(X0 * 2) - 1d) / (double)
       _CurrentNy;
13     Xf = (int)(X0 + (Xinc * (double)_CurrentNx));
14     Yf = (int)(Y0 + (Yinc * (double)_CurrentNy));
15
16     Rect Bounds = new Rect(new Point(0, 0), new Point(width - 1,
       height - 1));
17
18     rawStride = (width * pf.BitsPerPixel + 7) / 8;
19     pixelData = new byte[rawStride * height];
```

Listagem 3.12: Parte inicial do código do método *Refresh*

Inicialmente, no código apresentado anteriormente, os atributos *_CurrentNx* e *_CurrentNy* são atualizados com os valores de *_Nx* e *_Ny*, respetivamente. Esses atributos representam o número atual de valores nas coordenadas X e Y. Em seguida, os atributos *width* e *height*, que indicam o tamanho da área de desenho, são definidas com os valores mínimos de representação da *wafer*. Os atributos *X0* e *Y0* são responsáveis por indicar as coordenadas de início do desenho, considerando uma margem de exclusão nas bordas. Os atributos *Xinc* e *Yinc* são calculados para determinar a distância entre as coordenadas dos diferentes quadrados futuramente criados. Já os atributos *Xf* e *Yf* são calculados para determinar as coordenadas finais do desenho.

O atributo *Bounds* é criado para limitar o máximo tamanho da área do *TabControl*. De seguida, é calculado o valor da variável *rawStride* que representa o número de bytes por linha na matriz de pixeis. Na linha seguinte é então criado um *array* denominado de *pixelData* que tem como objetivo armazenar os valores dos pixeis da imagem. O tamanho do *array* é determinado pelo produto entre o *rawStride* e o *height*.

A seguir mostra-se o restante código do método *Refresh*, mais propriamente os *loops*.

```
for (int whiteIndx = 0; whiteIndx < rawStride * height; whiteIndx
    = whiteIndx + 3)
{
    pixelData[whiteIndx] = c.R;
    pixelData[whiteIndx + 1] = c.G;
    pixelData[whiteIndx + 2] = c.B;
}
for (int indX = 0; indX < _CurrentNx; indX++)
{
    for (int indY = 0; indY < _CurrentNy; indY++)
    {
        if (SquareCompletlyInsideCircle(X0 + indX * Xinc,Y0 + indY
     * Yinc,X0 + (indX + 1) * Xinc, Y0 + (indY + 1) * Yinc) ||
    _CurrSelectedDieData[indX, indY].DieWasMeasured)
        {
            _CurrSelectedDieData[indX, indY].DieInsideWafer=true;

            if (_CurrSelectedDieData[indX, indY].DieWasMeasured)
            {
                DrawFilledRectangle(X0 + indX * Xinc, Y0 + indY *
    Yinc, X0 + (indX + 1) * Xinc, Y0 + (indY + 1) * Yinc,
    _CurrSelectedDieData[indX,indY].color.GetBinColor().ToColor(),
    pixelData, rawStride);
            }
            DrawRectangle(X0 + indX * Xinc, Y0 + indY * Yinc, X0 +
     (indX + 1) * Xinc, Y0 + (indY + 1) * Yinc, Colors.Black,
    pixelData, rawStride);
        }
        else
        {
            _CurrSelectedDieData[indX, indY].DieInsideWafer =
    false;
        }
    }
}
    ellipse(Bounds, Colors.Red, pixelData, rawStride);
    Notch(Bounds, Colors.Red, pixelData, rawStride);

    Render();
}
```

Listagem 3.13: Parte final do código do método *Refresh*

O primeiro *loop for* percorre todos os pixeis da imagem atribuindo os valores "R"(vermelho), "G"(verde) e "B"(azul) fazendo com que toda a área seja pintada a branco, pela junção das três cores, para evitar que futuramente a área externa à *wafer* fique pintada a preto. O segundo *loop* é iniciado para percorrer todos os elementos

em X e Y. Para cada elemento, é verificado se ele está completamente dentro de um círculo ou se já foi desenhado. Se uma destas condições for verdadeira, o atributo *DieInsidewafer* do elemento é definido como verdadeiro. Além disso, se o elemento já foi desenhado, um retângulo preenchido é desenhado na área correspondente usando a cor associada a esse elemento. Para delimitar e salientar melhor os retângulos é também desenhado os seus limites a preto.

Após o *loop*, são chamados os métodos *ellipse* e *notch* para desenhar uma elipse vermelha e um *notch* na imagem. O *notch* desempenha um papel crucial, pois é responsável por fornecer informações sobre a orientação da *wafer*. Por fim, é chamado o método *Render* para exibir a imagem atualizada. Na figura abaixo, é apresentado o gradiente de cores desejado para um teste, localizado no *TabControl*. Também é possível observar a presença de vários separadores, que indicam a capacidade de abrir simultaneamente mais do que um teste.

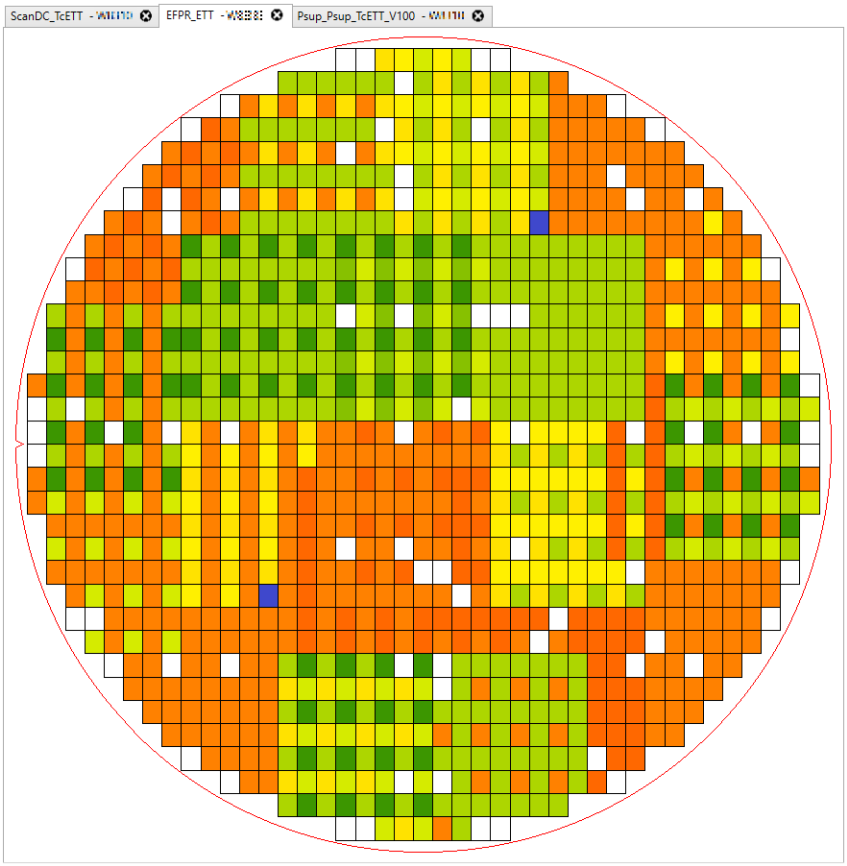


Figura 3.10: Exemplo de um *wafermap* [1]

[1]Por motivos de confidencialidade, a figura encontra-se propositadamente desfocada.

A implementação do *TabControl* tem como objetivo melhorar a usabilidade e eficiência da aplicação, permitindo ao utilizador explorar e comparar *wafermaps* de forma conveniente e intuitiva proporcionando uma experiência mais completa aos utilizadores.

3.3.6 Barras

A implementação de barras numa interface complementa a informação melhorando assim a experiência do utilizador. As barras basicamente oferecem acesso rápido a funcionalidades importantes ou informações adicionais. Elas contribuem para uma experiência mais intuitiva e eficiente, garantindo que os utilizadores possam melhorar a sua experiência com a interface.

3.3.6.1 Barra Superior

Foi adicionada uma barra superior à aplicação, introduzindo recursos para complementar e melhorar a experiência dos utilizadores. A barra superior é constituída por dois menus principais: *File* e *Help*.

No menu *File*, foram incluídas várias opções que oferecem controlo sobre os ficheiros e os separadores abertos no *TabControl*. As opções são as seguintes:

- *Open File*: À semelhança do botão, esta opção, permite aos utilizadores selecionar e abrir um ficheiro específico;
- *Copy Image To Clipboard*: Oferece a capacidade de copiar o *wafermap* presente na *TabControl* para a área de transferência, facilitando a sua partilha ou uso posterior;
- *Close Current Tab*: Fecha o separador atualmente selecionado, permitindo aos utilizadores gerir os seus separadores de forma mais eficiente;
- *Close All Open Tabs*: Fecha todos os separadores aberto na aplicação de uma só vez, possibilitando uma limpeza rápida e organização dos separadores;
- *Exit*: Encerra completamente a aplicação.

A figura seguinte mostra o menu *File* da barra superior.

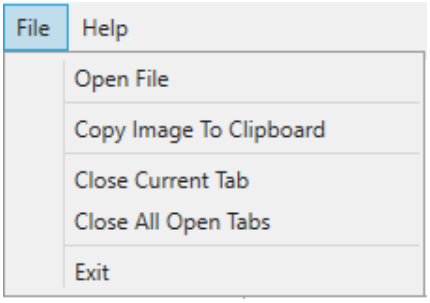


Figura 3.11: Menu *File*

Já no menu *Help* é possível encontrar as seguintes opções.

- *Help*: Ao selecionar esta opção, uma nova janela é aberta, fornecendo informações detalhadas e úteis sobre como utilizar efetivamente a aplicação. Esta secção de ajuda tem como objetivo orientar os utilizadores a aproveitarem ao máximo os recursos disponíveis;
- *About*: Esta opção, tal como a anterior, também abre uma nova janela. No entanto, desta vez, é realçada a informação de que a aplicação foi desenvolvida no âmbito da unidade curricular de Tese/Dissertação.

A figura seguinte mostra o menu *Help* da barra superior.

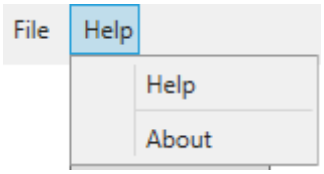


Figura 3.12: Menu *Help*

A implementação da barra superior enriquece a aplicação, fornecendo recursos e informações relevantes para os utilizadores da aplicação.

3.3.6.2 Barra Inferior

Foi implementada uma barra inferior para melhorar a experiência do utilizador, oferecendo informações adicionais relativas ao *wafermap*. A barra inferior é composta por quatro campos diferentes que apresentam informações para o utilizador.

No primeiro campo baseado no nome do último ficheiro aberto é apresentado o nome da *wafer*. Através deste campo, o utilizador não tem a necessidade de se lembrar qual foi o ficheiro que carregou para a aplicação, como se exemplifica na figura seguinte.

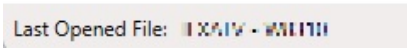


Figura 3.13: Primeiro campo da barra inferior [2]

A segunda informação exibida na barra inferior corresponde ao número de quadrados associados a cada cor presente no *wafermap*. Esta informação revela-se útil para complementar a análise e controlo da qualidade dos semicondutores. Ao avaliar o número de quadrados de cada cor, o utilizador pode identificar de forma rápida eventuais desvios em relação à distribuição esperada, possibilitando a deteção de possíveis defeitos na produção. Em seguida, é possível observar na figura o campo referente a esta informação.

Figura 3.14: Segundo campo da barra inferior

A terceira informação exibida na barra inferior é essencial para a análise de um *wafermap*. Ela inclui os limites aceites pelo teste selecionado. Com base na posição do rato, exibe as coordenadas e o valor correspondente do quadrado em específico. Ter acesso às coordenadas e aos valores correspondentes de quadrados na barra inferior permite ao utilizador acompanhar os valores dos resultados, proporcionando uma visão mais clara e imediata, facilitando a interpretação dos mesmos. Na figura seguinte pode ser visualizado este campo.

Figura 3.15: Terceiro campo da barra inferior

[2]Por motivos de confidencialidade, a figura encontra-se propositadamente desfocada.

Por fim, o último campo da barra inferior foi concebido a pensar nos utilizadores que possuam uma perceção cromática diferente, ou seja, que não consigam visualizar as cores da mesma forma que os outros utilizadores. Com esse intuito, o último campo exibe em forma de texto a cor do quadrado onde o rato se encontra. Essa informação auxiliará os utilizadores com uma perceção cromática diferente a compreender como o gradiente de cores está disposto. As cores presentes nesse gradiente são: "Green"(Verde), "Yellow"(Amarelo), "Orange"(Laranja) e "Blue - Out of Range"(Azul - Fora do Intervalo). Na figura seguinte, é possível observar o referido campo.

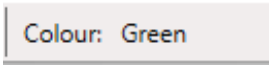


Figura 3.16: Quarto campo da barra inferior

3.4 Gestão de Dados em Memória

A gestão de memória é crucial em qualquer *software*, independente do seu tamanho ou complexidade. Uma gestão de memória eficiente otimiza a utilização dos recursos disponíveis e ajuda a evitar problemas como fugas de memória, fragmentação e falhas no sistema.

3.4.1 Aquisição de Dados

Os dados do ficheiro Excel são obtidos através de testes realizados na *wafer*. Como explicado anteriormente, a *wafer* é uma placa plana de material semicondutor. Nesse contexto, os *sites* referem-se a agulhas que entram em contacto com a *wafer* durante os testes. Cada vez que esses *sites* tocam na *wafer*, obtêm-se os dados que são posteriormente registados no Excel. Esse processo de contacto entre os *sites* e a *wafer* é conhecido como *touchdown*. Para além dos valores obtidos nos testes, também são registadas as coordenadas cartesianas correspondentes a cada ponto de teste.

Após a escolha do ficheiro Excel na aplicação, foi possível realizar a leitura e a divisão dos dados obtidos em três estruturas distintas sendo elas o *testheader*, o *devicedata* e o *sitedata*.

A estrutura *testheader*, como o próprio nome indica, tem a função de armazenar informações relacionadas ao cabeçalho. O cabeçalho contém dados importantes, como os nomes dos diferentes testes realizados, os identificadores únicos atribuídos a

cada teste, o intervalo de valores no qual os resultados devem estar para garantir que o teste seja considerado bem-sucedido, o número total de quadrados testados e, por fim, a quantidade de quadrados que não foram bem-sucedidos no teste. Além disso, o *testheader* também indica a unidade de medida dos diferentes testes, fornecendo contexto adicional para a interpretação dos resultados. De seguida, está presente no extrato de código a declaração dos atributos que guardam toda esta informação.

```
1  public class testheader
2  {
3      private String[] testname = new String[Constants.maxNTest];
4      private int [] testnumber = new int[Constants.maxNTest];
5      private double?[] lowlimit = new double?[Constants.maxNTest];
6      private double?[] highlimit = new double?[Constants.maxNTest];
7      private String[] unit = new String[Constants.maxNTest];
8      private int?[] executed = new int?[Constants.maxNTest];
9      private int?[] nfails = new int?[Constants.maxNTest];
10 }
```

Listagem 3.14: Atributos da classe *testheader*

Esta estrutura, criada com o nome *testheader*, tem como objetivo armazenar os valores obtidos na leitura do ficheiro Excel e foi declarada como *public* para que a informação por ela armazenada possa ser acedida em qualquer parte do código.

A estrutura *devicedata* tem a utilidade de armazenar os valores de cabeçalho relacionados ao dispositivo de medição, como o número do *site*, as coordenadas cartesianas e o número de *touchdown*.

Estes valores são essenciais para fornecer informações de identificação da máquina, permitindo a identificação do respetivo *die* (*chip* individual presente na *wafer*). Ao armazenar essas informações na classe *devicedata*, é possível associar cada conjunto de dados de medição a uma máquina específica, facilitando a rastreabilidade e a análise dos resultados obtidos nos vários *dies* presentes nas *wafers*. De seguida está presente no extrato de código seguinte a declaração da estrutura *devicedata* e dos seus principais atributos.

```
1 public class devicedata
2 {
3     private int[] site= new int[Constants.maxDiesPerWafer];
4     private int[] xcoord= new int[Constants.maxDiesPerWafer];
5     private int[] ycoord= new int[Constants.maxDiesPerWafer];
6     private int[] td= new int[Constants.maxDiesPerWafer];
7 }
```

Listagem 3.15: Variáveis da classe *devicedata*

Por fim, a estrutura *testdata* foi projetada para lidar com o armazenamento de um grande número de valores provenientes dos diversos testes efetuados, facilitando o acesso e a manipulação desses dados para realizar a análise desejada. Na próxima figura está presente o atributo declarado como *tresults*.

```
1 public class testdata
2 {
3     private List<List<double?>> tresults = new List<List<double
      ?>>();
4 }
```

Listagem 3.16: Atributo da estrutura *testdata*

Para não existir uma acumulação de dados nas estruturas é necessário libertar a memória ocupada anteriormente. Em C#, não é necessário fazer a limpeza da memória manualmente, pois o *garbage collector* do *.NET Framework* é responsável por gerir automaticamente a memória.

Para que a memória seja automaticamente gerida basta criar um objeto usando o operador *new*, assim que não existam mais referências ao objeto, ele é automaticamente libertado da memória. De seguida, é possível visualizar no extrato de código, a utilização do operador *new* sempre que se clica no botão *Open File* para que exista uma gestão de dados em memória eficiente sem acumular demasiados dados nas estruturas criadas.

```
1 mtestheader = new testheader();
2 mdevicedata = new devicedata();
3 msitedata = new sitedata();
```

Listagem 3.17: Atributo da estrutura *testdata*

3.5 Correção e Prevenção de Erros

A prevenção de erros desempenha um papel fundamental quando se procura criar aplicações mais eficientes e confiáveis. Ao longo do desenvolvimento de um *software*, é essencial antecipar e mitigar possíveis falhas que possam comprometer a funcionalidade, segurança e usabilidade do programa.

Nesta secção, serão abordadas as estratégias e boas práticas implementadas para tornar a aplicação mais robusta, corrigindo desde logo quatro erros identificados:

1. Restrição nos ficheiros Excel inseridos: Foi implementada uma restrição para permitir apenas a inserção de ficheiros Excel correspondentes às *wafers* compatíveis com a versão de *software* desenvolvida. Isso garante que apenas os ficheiros compatíveis sejam inseridos, evitando problemas de incompatibilidade e assegurando a integridade dos dados;

2. Limite de valores inseridos nos percentis: Para o cálculo do percentil, é apenas permitido inserir valores no intervalo de 0 a 100. Esta limitação evita erros no cálculo do percentil e assegura que os resultados sejam precisos e coerentes;

3. Restrição nas caixas de pesquisa: Foi definido um limite para o número de caracteres aceites, evitando o sobrecarregamento do *software* e garantindo a correta interpretação dos dados inseridos. Além disso, na caixa de pesquisa de números de teste, foi implementada uma restrição que apenas permite a introdução de números;

4. Limite de abertura de separadores no *TabControl*: Para evitar uma sobrecarga excessiva na aplicação, foi estabelecido um limite máximo de oito separadores abertos simultaneamente no *TabControl*. Essa limitação contribui para uma melhor organização e legibilidade das informações apresentadas, tornando a experiência do utilizador mais fluída.

Para complementar as correções dos erros mencionados, foi implementado um recurso adicional que exibe um *pop-up* com uma mensagem de texto sempre que o utilizador não cumpra as restrições estabelecidas, como por exemplo, inserir valor fora do intervalo aceite no percentil. Esse *pop-up* funciona como um alerta, informando o utilizador sobre o erro cometido e fornecendo orientações sobre como corrigi-lo.

Desta forma, para além das limitações e restrições incorporadas nos elementos de interação da aplicação, o *feedback* imediato através do *pop-up* contribui para uma melhor experiência e facilita a introdução correta de dados. Essa abordagem proativa promove a usabilidade e eficiência do *software*, permitindo que o utilizador compreenda rapidamente o que precisa de ser ajustado para prosseguir com sucesso nas suas interações.

Capítulo 4

Demonstração e análise de resultados

Neste capítulo, numa primeira parte, será realizada uma demonstração detalhada do funcionamento da aplicação. Posteriormente, serão apresentados os resultados obtidos a partir de um inquérito, presente no anexo A, realizado sobre a aplicação desenvolvida após a sua utilização por um grupo de teste. Serão analisadas as respostas dos utilizadores ao inquérito, abordando aspetos como a facilidade de utilização, a interação com a aplicação e de que forma esta melhorou a análise dos dados paramétricos. A amostra do inquérito incluiu cinco engenheiros que possuem familiaridade com a análise desse tipo de dados paramétricos.

4.1 Demonstração

Esta seção tem como objetivo a apresentação e demonstração do *software* desenvolvido. Será fornecida uma descrição detalhada das funcionalidades do *software*, acompanhada por imagens ilustrativas que demonstrarão todas as características do programa.

Em primeiro lugar, serão destacados os principais passos e procedimentos necessários para obter o gradiente no formato de *wafermap*. Na figura seguinte, é possível observar a interface inicial, a qual se apresenta vazia nos diversos campos (com exceção dos valores padrão utilizados no cálculo dos percentis).

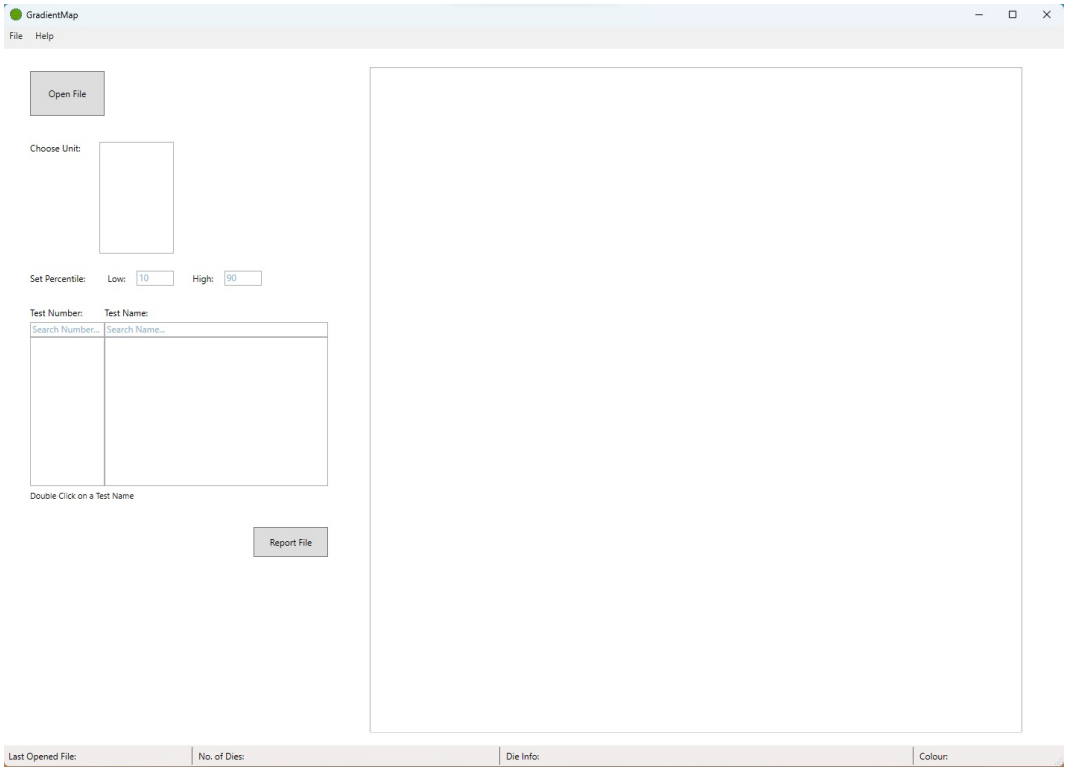


Figura 4.1: Interface inicial

O primeiro passo consiste em pressionar o botão *Open File*. Ao fazê-lo, será apresentada uma janela de seleção de ficheiros, permitindo a navegação pelo sistema de ficheiros do computador para localizar o ficheiro desejado. Esta funcionalidade facilita o processo de importação de dados para a aplicação.

Após selecionar o ficheiro, a aplicação iniciará o processo de leitura do mesmo. No final deste processo, serão apresentadas na *listbox* de parâmetros, todas as unidades guardadas no ficheiro associadas a parâmetros testados.

Esta funcionalidade serve como um primeiro filtro para lidar com a grande quantidade de testes presentes num único ficheiro de dados. A *listbox* exibe agora uma lista organizada das unidades associadas aos parâmetros identificados, permitindo que o utilizador selecione e interaja com cada um deles de acordo com as suas necessidades. Ao analisar a figura seguinte, é possível notar que, nesse momento específico, a *listbox* é a única área com dados visíveis, aguardando as próximas interações do utilizador.

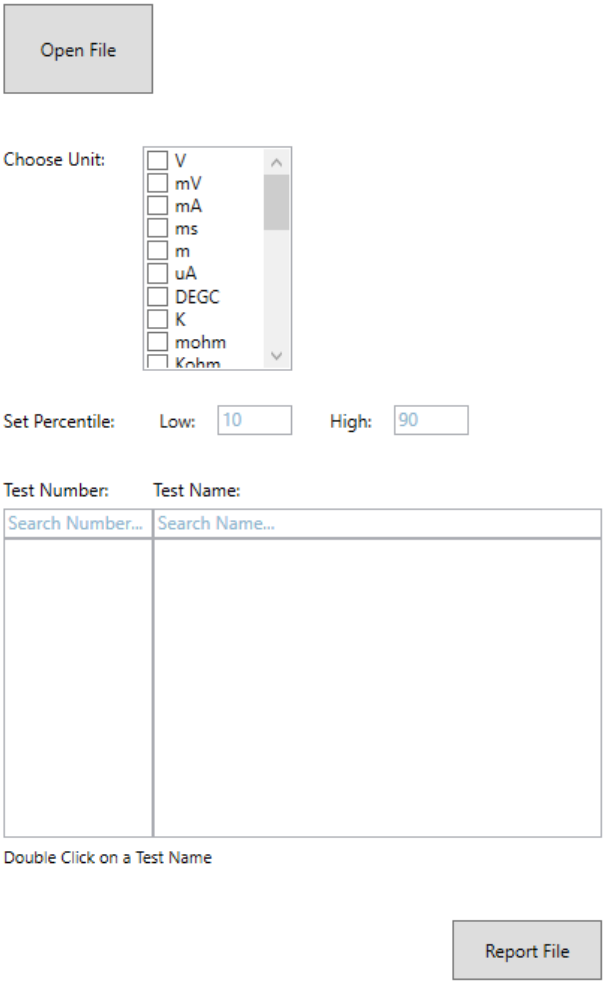


Figura 4.2: Interface apenas com as unidades na *listbox*

Após o procedimento anterior, é agora necessário selecionar um ou mais parâmetros para que sejam apresentados em baixo tanto os números como os nomes dos testes correspondestes.

Ao selecionar as opções desejadas, a interface irá atualizar dinamicamente na *listbox*, apresentando os testes associados à seleção.

Na próxima figura, é percetível que, devido à seleção de dois parâmetros, as *listbox* apresentam agora informações relevantes, permitindo assim, um acesso mais preciso aos números ou nomes dos testes.

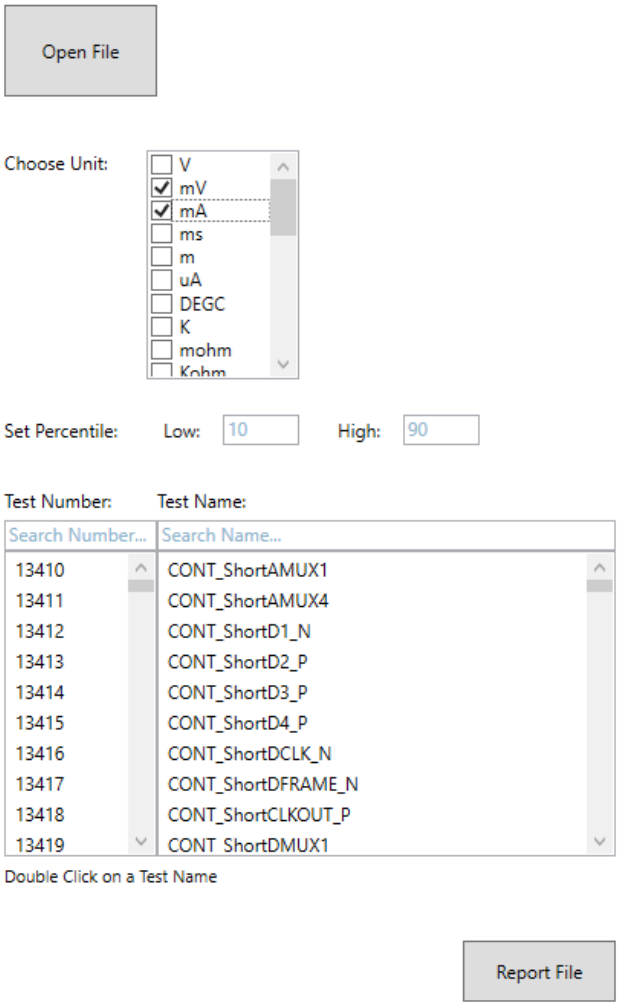


Figura 4.3: Interface com unidades selecionadas

A fim de desenhar o gradiente de cores e apresentá-lo na forma de *wafermap*, é necessário seguir um procedimento simples. Primeiro, para gerar o *wafermap* com o gradiente de cores é preciso escolher um teste específico na *listbox* correspondente e depois fazer um clique duplo no seu nome. Ao se realizar esta ação, o *wafermap* desejado será imediatamente exibido no *TabControl*, localizado no lado direito da interface.

Esta funcionalidade permite a visualização clara e detalhada do gradiente de cores correspondente ao teste selecionado. O *TabControl* oferece uma maneira organizada de apresentar diferentes *wafermaps*, garantindo que o utilizador consegue alterar facilmente entre eles.

Na figura seguinte está presente um exemplo de um gradiente de cores no formato de *wafermap* onde é possível reparar que na barra inferior está presente a que *wafer*

pertence ao último ficheiro aberto, o número de quadrados de cada cor, a informação relativa a um quadrado específico e por fim, uma legenda da cor do quadrado para ajudar o utilizador que possua uma banda cromática diferente.

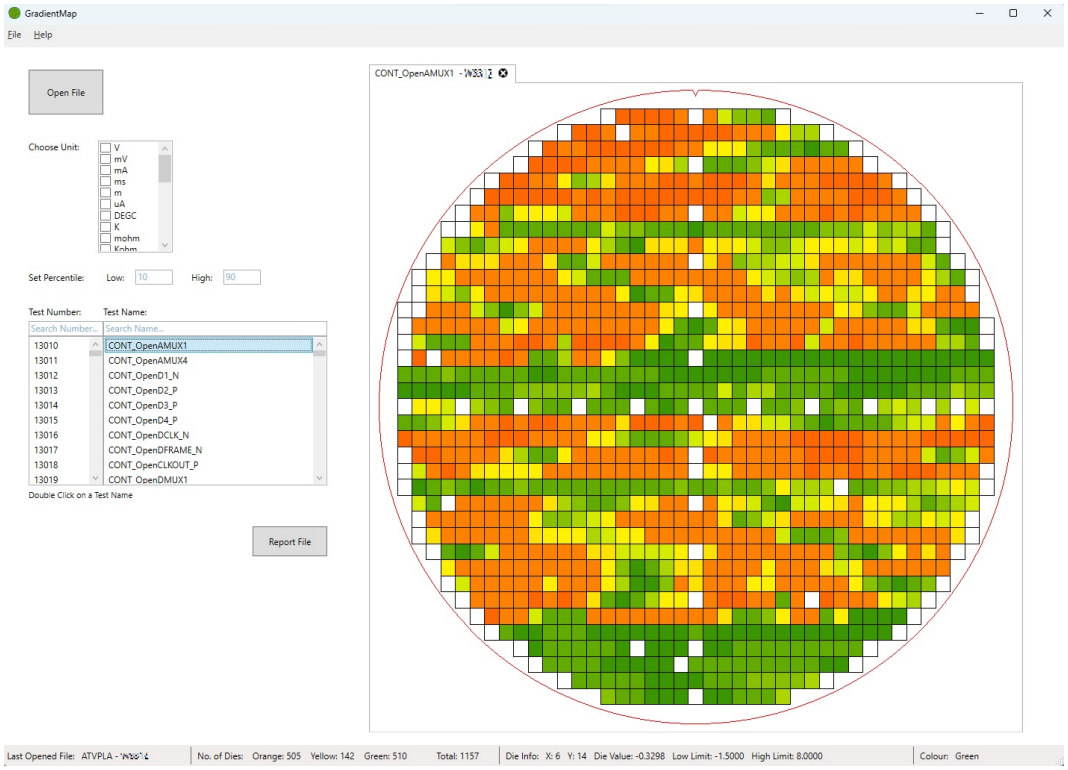


Figura 4.4: Interface com a *wafer* impressa no *TabControl* [1]

Por fim, para permitir que o utilizador guarde o *wafermap* presente no *TabControl*, foi implementado um botão *Report File*. Ao clicar neste botão, é criado automaticamente um relatório num ficheiro em formato PDF com as informações essenciais da análise realizada. Nessas informações pode-se ler o nome do ficheiro, o nome da *wafer*, o percentil utilizado para definir os diferentes intervalos de valores e por fim, a imagem do *wafermap*. De seguida, é apresentada uma figura que mostra como a informação é guardada no documento PDF.

[1] Por motivos de confidencialidade, a figura encontra-se propositadamente desfocada.

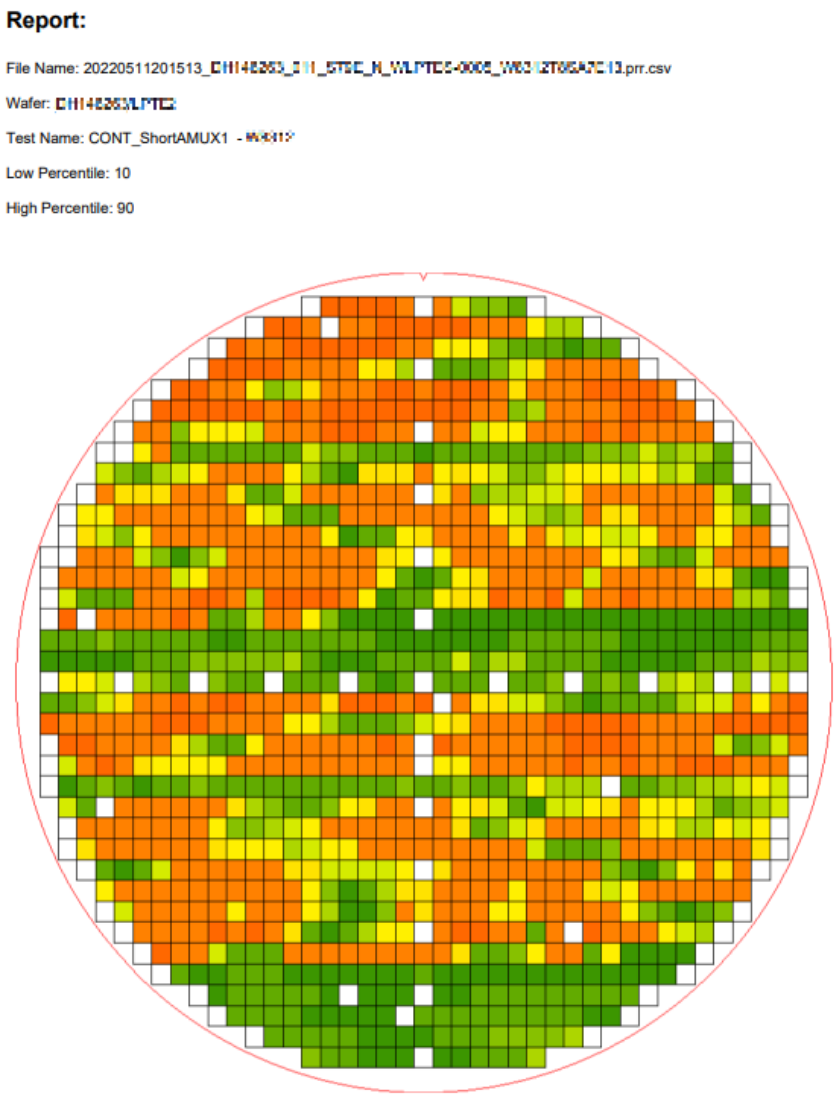


Figura 4.5: Informação presente no PDF gerado [2]

Esta opção de exportação em formato PDF foi criada com o intuito de oferecer uma solução simples e conveniente para que os utilizadores possam preservar e compartilhar o *wafermap*. Este botão garante que o ficheiro resultante contenha todas as informações relevantes definidas na especificação da empresa, relativas ao teste selecionado.

4.2 Apresentação das perguntas

Todas as perguntas, à exceção da última, foram formuladas de modo a serem respondidas numa escala de *Likert* de 1 a 5. Optou-se por utilizar este tipo de escala

[2]Por motivos de confidencialidade, a figura encontra-se propositadamente desfocada.

com o objetivo de reduzir o tempo de resposta do utilizador e facilitar a análise posterior dos dados.

De seguida, são apresentadas as perguntas que constituíram o inquérito:

1. Como avalia o aspeto (*layout*) da interface;
2. Como avalia a facilidade de utilização (interação com os menus);
3. Como avalia os tempos de resposta (desempenho) da aplicação;
4. Avalie a facilidade de escolha de um ficheiro Excel de um teste paramétrico;
5. Avalie de que forma a escolha das unidades da lista apresentada é adequada;
6. Como avalia a possibilidade de alterar os valores dos percentis;
7. Como avalia a utilidade das caixas de pesquisa (*search bar*), *Test Number* e *Test Name*;
8. Como avalia a apresentação dos testes disponíveis, na listagem interativa lateral (*Test Number* e *Test Name*);
9. Como avalia a facilidade de gerar um *wafermap*;
10. Avalie de que forma o gradiente de cor é apresentado de forma clara e adequada no *wafermap*;
11. Como avalia a importância da possibilidade de abrir vários separadores (*wafermap*) correspondentes a diferentes testes;
12. Como avalia a possibilidade de poder guardar em PDF o estudo sobre um teste (*wafermap*);
13. Como avalia a importância da informação relativa ao último ficheiro aberto, presente no lado esquerdo da barra inferior;
14. Como avalia a importância dos dados obtidos na barra inferior quando se move o rato por cima do *wafermap* (coordenadas, valor do teste e cor do *die*);
15. Como avalia a facilidade em encontrar as informações pretendidas, para uma análise de um teste;
16. Classifique a adequação das funcionalidades desenvolvidas, face aos objetivos iniciais do utilizador;
17. Classifique de que forma a aplicação melhorou a análise dos dados paramétricos de testes;
18. De forma geral como classifica a aplicação desenvolvida;

19. Existe algum aspeto específico que gostaria de mencionar em relação à aplicação desenvolvida?

4.3 Análise de Dados

Inicialmente colocou-se cada candidato a utilizar a ferramenta, chamando à atenção para os aspetos que importava estudar. Com o intuito de evitar qualquer dúvida por parte dos inquiridos em relação às perguntas propostas, foi disponibilizado uma imagem da interface juntamente com o inquérito. Essa imagem, acompanhada por legendas correspondentes a cada pergunta, será apresentada na secção das perguntas específicas. Dessa forma, busca-se proporcionar uma melhor compreensão e clareza aos participantes. De seguida, apresenta-se uma figura da interface ainda sem legendas.

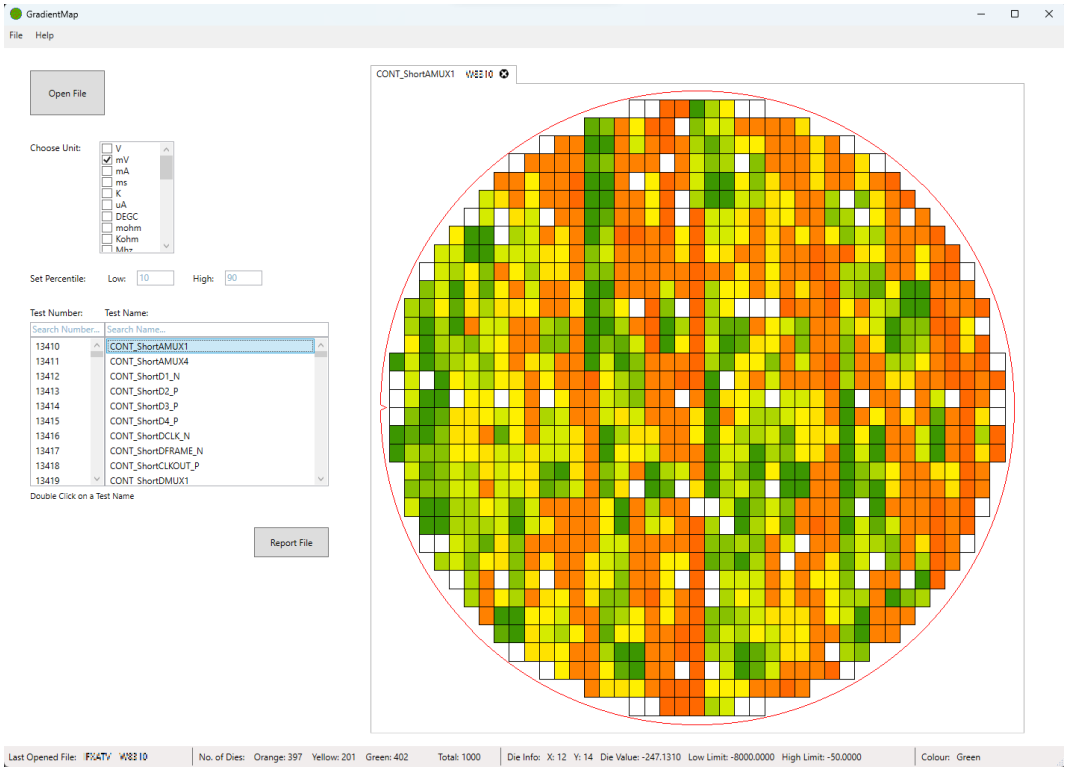


Figura 4.6: Interface da aplicação [3]

Para analisar os dados, optou-se por organizá-los em duas componentes, sendo a primeira referente a perguntas gerais de utilização da aplicação e a segunda relacionada a perguntas mais específicas sobre os elementos da interface.

[3]Por motivos de confidencialidade, a figura encontra-se propositadamente desfocada.

4.3.1 Perguntas Gerais

Para avaliar as perguntas gerais acerca da aplicação optou-se por criar um gráfico de barras com todas as respostas relacionadas com as questões que abordam a interface no seu geral. Na figura seguinte, é possível visualizar o gráfico correspondente a esse conjunto de perguntas.

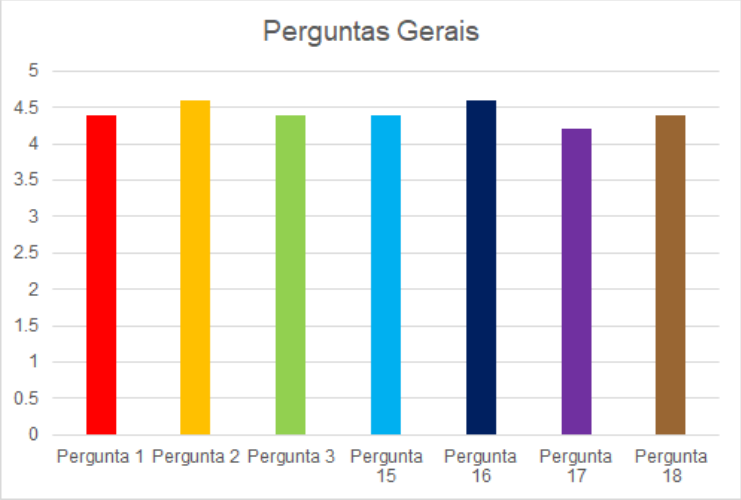


Figura 4.7: Perguntas Gerais

Como se pode ver pela legenda, cada barra corresponde a uma pergunta das enumeradas em cima na secção 4.2.

Após analisar o gráfico, é possível constatar que a maioria dos inquiridos avaliou positivamente o aspeto da interface, destacando que o design visual é agradável (P1). Além disso, demonstraram satisfação com a facilidade de utilização e interação com os menus oferecidos, considerando a aplicação intuitiva e de fácil navegação (P2).

No que diz respeito ao desempenho da aplicação, os participantes avaliaram os tempos de resposta como adequados (P3). Os inquiridos também reportaram que foi fácil encontrar as informações desejadas para a análise de testes, o que demonstra que a organização e a estrutura da aplicação facilitam a localização das interações relevantes (P17).

Em relação à adequação das funcionalidades desenvolvidas, os inquiridos responderam de forma muito positiva, indicando que as mesmas satisfazem as necessidades pretendidas pelos utilizadores (P18).

Adicionalmente, a aplicação contribuiu para melhorar a análise dos dados paramétricos de testes (P19), embora essa área tenha obtido uma pontuação ligeiramente inferior em comparação com as outras, continuando, no entanto, a apresentar uma média superior a quatro, 4.2 ± 0.4. De forma geral, os participantes classificaram a

aplicação como satisfatória, demonstrando uma impressão positiva sobre o trabalho desenvolvido.

4.3.2 Perguntas Específicas

Antes de iniciar a análise dos dados obtidos, será exibida uma imagem com a legenda na interface, mostrando a correspondência entre cada pergunta e elemento da interface. Esta figura tem como objetivo facilitar a compreensão e proporcionar uma visão abrangente do conteúdo abordado.

Figura 4.8: Interface com legenda [4]

Para apresentar a avaliação das caraterísticas mais específicas acerca da aplicação optou-se por criar igualmente um gráfico de barras. Neste gráfico são apresentadas as respostas das perguntas relacionadas com elementos específicos da interface. Na figura seguinte, é possível visualizar o gráfico correspondente a esse grupo de perguntas.

[4]Por motivos de confidencialidade, a figura encontra-se propositadamente desfocada.

Figura 4.9: Perguntas Específicas

Para analisar este gráfico, serão abordadas as perguntas de forma individualmente, uma vez que se referem a elementos da interface.

- Pergunta 4:

  A facilidade de escolha de um ficheiro Excel de um teste paramétrico recebeu uma avaliação extremamente positiva, com uma pontuação média de 4.8 ± 0.4. Isso reflete a opinião dos utilizadores, que consideraram essa tarefa particularmente fácil e intuitiva. A interface da aplicação proporciona uma experiência sem complicações, permitindo que os utilizadores selecionem um ficheiro Excel com facilidade e sem dificuldades adicionais.

  O botão de seleção de ficheiro desempenha um papel essencial ao simplificar o processo para os utilizadores. Com uma interação simples, os utilizadores podem navegar e selecionar o ficheiro Excel desejado sem a necessidade de procedimentos complexos ou passos adicionais.

  A avaliação positiva da facilidade de escolha de um ficheiro Excel é um testemunho do *design* intuitivo da interface da aplicação e da atenção dedicada ao desenvolvimento de uma funcionalidade que atende às necessidades dos utilizadores de forma eficiente e eficaz.

- Pergunta 5:

  A avaliação da forma como a escolha das unidades da lista apresentada foi adequada revela uma resposta altamente positiva, com uma pontuação média de 4.8 ± 0.4. Isso indica que os utilizadores consideraram as opções de seleção na lista como relevantes e apropriadas para a análise dos testes paramétricos.

A lista de parâmetros é apresentada de forma visualmente organizada, permitindo que os utilizadores identifiquem e selecionem facilmente os parâmetros desejados. Essa abordagem facilita a navegação e a seleção precisa das unidades relevantes para a análise dos testes paramétricos.

Esta avaliação positiva reforça o cuidado colocado no *design* da interface, garantindo que as opções de seleção estejam bem alinhadas com as necessidades dos utilizadores. A escolha adequada das unidades na lista proporciona uma experiência intuitiva e eficiente, contribuindo para a qualidade geral da aplicação e para a satisfação dos utilizadores.

- Pergunta 6:

  A possibilidade de alterar os valores dos percentis recebeu uma avaliação positiva, com uma pontuação média de 4.6 ± 0.5. Isso indica que os utilizadores consideraram essa funcionalidade fácil de utilizar e valorizaram a capacidade de personalização oferecida.

  A interface da aplicação proporciona uma forma intuitiva e acessível de modificar os valores dos percentis. Os utilizadores podem facilmente ajustar os percentis de acordo com as suas preferências, o que contribui para uma experiência positiva e personalizada na aplicação. Além disso, é importante destacar a clareza na perceção dos valores predefinidos dos percentis. Os utilizadores conseguem compreender facilmente quais são os valores de referência iniciais e têm a flexibilidade de ajustá-los conforme necessário.

  Essa facilidade de uso e a capacidade de personalização dos valores dos percentis promovem uma maior usabilidade da aplicação, permitindo aos utilizadores adaptarem as configurações às suas necessidades específicas. Essa funcionalidade contribui para uma experiência mais satisfatória e eficiente na análise dos dados paramétricos.

- Pergunta 7:

  A caixa de pesquisa (*search box*) para filtrar os testes por *Test Number* ou *Test Name* recebeu uma avaliação positiva, obtendo uma pontuação média de 4.4 ± 0.5. Isso indica que os utilizadores consideraram essa caixa de pesquisa útil e eficaz para encontrar os testes desejados.

  A avaliação positiva da caixa de pesquisa ressalta a sua utilidade na interface da aplicação, demonstrando que os utilizadores a consideraram uma ferramenta eficiente para a pesquisa e filtragem dos testes. A presença dessa caixa de pesquisa simplifica o processo de localização e acesso aos testes desejados, otimizando a experiência dos utilizadores na aplicação.

- Pergunta 8:

  A apresentação dos testes disponíveis na listagem interativa lateral, com as informações de *Test Number* e *Test Name*, recebeu uma avaliação positiva, obtendo uma pontuação média de 4.4 ± 0.5. A listagem dos testes proporciona uma visualização clara e organizada, permitindo aos utilizadores percorrer facilmente a lista e encontrar os testes com base no *Test Number* ou *Test Name*. Essa estrutura ajuda a melhorar a usabilidade da aplicação, uma vez que os utilizadores podem localizar rapidamente os testes desejados sem esforço adicional.

  A avaliação positiva da apresentação dos testes disponíveis reforça o cuidado em projetar uma interface que atenda às necessidades dos utilizadores e facilite a busca e seleção dos testes. A visualização clara e organizada dos testes na listagem lateral promove a eficiência e a satisfação dos utilizadores, contribuindo para a qualidade geral da aplicação.

- Pergunta 9:

  A facilidade de gerar um *wafermap* recebeu uma avaliação positiva, com uma pontuação média de 4.4 ± 0.5. Essa avaliação reflete a perceção dos utilizadores de que essa tarefa é relativamente fácil de executar na aplicação.

  A funcionalidade de gerar um *wafermap* é projetada para proporcionar uma experiência intuitiva e eficiente aos utilizadores. Através de uma abordagem simplificada e acessível, os utilizadores podem realizar essa tarefa de forma rápida e sem complicações.

  A possibilidade de gerar um *wafermap* de forma fácil e eficiente contribui para a análise dos dados paramétricos de forma visual e aprofundada, enriquecendo a experiência dos utilizadores na aplicação.

- Pergunta 10:

  A apresentação do gradiente de cor no *wafermap* foi avaliada de forma positiva, obtendo uma média de resposta de 4.6 ± 0.5. Essa avaliação reflete a perceção dos utilizadores de que a representação do gradiente de cor é clara e adequada para mostrar as diferentes variações nos valores.

  O gradiente de cor é apresentado de forma visual, permitindo uma compreensão fácil e intuitiva das informações contidas no *wafermap*. Os utilizadores podem identificar visualmente os padrões, tendências ou áreas de interesse com base nas variações de cor, o que facilita a tomada de decisões e o estudo dos resultados.

  A avaliação positiva da clareza e adequação da apresentação do gradiente de cor no *wafermap* destaca o cuidado em projetar uma visualização que atenda

às necessidades dos utilizadores e otimize a interpretação dos dados paramétricos. Essa representação visual eficiente contribui para a qualidade geral da aplicação, proporcionando aos utilizadores uma experiência mais satisfatória e informada.

- Pergunta 11:

  A importância da possibilidade de abrir vários separadores correspondentes a diferentes testes e/ou *wafers* recebeu uma avaliação positiva, com uma média de resposta de 4.6 ± 0.5. Essa avaliação reflete o reconhecimento dos utilizadores quanto à utilidade e relevância dessa funcionalidade na aplicação. A capacidade de abrir múltiplos separadores permite aos utilizadores realizarem comparações e análises simultâneas entre os resultados de diferentes testes e/ou *wafers*. Essa funcionalidade facilita a visualização e a compreensão das variações e padrões existentes nos dados paramétricos, contribuindo para uma análise mais abrangente e detalhada.

  A avaliação positiva da importância da possibilidade de abrir vários separadores ressalta a relevância dessa funcionalidade na aplicação. Ao fornecer aos utilizadores a flexibilidade de analisar e comparar simultaneamente os resultados de diferentes testes e/ou *wafers*, a aplicação busca atender às necessidades dos utilizadores e promover uma análise mais completa e precisa dos dados paramétricos.

- Pergunta 12:

  A possibilidade de guardar o estudo sobre um teste (*wafermap*) em formato PDF recebeu uma avaliação relativamente baixa, com uma média de resposta de 3.0 ± 0.6. Isso sugere que os utilizadores consideraram essa funcionalidade menos relevante em comparação com outras funcionalidades disponíveis na aplicação.

  O resultado obtido pode ser atribuído ao facto de que a opção de guardar o estudo em PDF ter sido uma implementação adicional, não estando inicialmente presente nos requisitos da empresa.

  Embora a avaliação tenha sido mais baixa, é necessário considerar que a utilidade e relevância dessa funcionalidade pode variar de acordo com as necessidades específicas dos utilizadores e dos contextos de utilização. É possível que, para certos cenários de uso, a capacidade de guardar o estudo em PDF seja considerada valiosa.

- Pergunta 13:

  A importância da informação relativa ao último ficheiro aberto, presente no lado esquerdo da barra inferior, foi avaliada de forma positiva, com uma pontuação média de 4.2 ± 0.4. Isso sugere que os utilizadores consideraram essa informação relevante e útil.

  A presença da informação na barra inferior permite aos utilizadores ter conhecimento rápido do último ficheiro Excel aberto. Ao exibir esta informação de forma proeminente na interface, a aplicação demonstra uma preocupação em facilitar a experiência do utilizador, oferecendo uma visão instantânea do contexto de trabalho mais recente.

- Pergunta 14:

  A importância dos dados obtidos na barra inferior ao mover o rato sobre o *wafermap* foi avaliada de forma muito positiva, com uma pontuação média de 4.6 ± 0.5. Isso indica que os utilizadores consideraram essas informações relevantes e úteis durante a interação com o *wafermap*.

  Os dados apresentados, como as coordenadas, o valor do teste e a cor do *die*, foram considerados essenciais para uma análise mais detalhada e precisa dos resultados. Permitem aos utilizadores obter uma compreensão imediata das características de cada quadrado constituinte do *wafermap*, facilitando a identificação de padrões, tendências ou áreas de interesse. Ao fornecer esses dados na barra inferior enquanto o utilizador move o rato sobre o *wafermap*, a aplicação demonstra uma abordagem de design cuidada, que visa disponibilizar informações contextuais relevantes de forma intuitiva e eficiente.

  Assim, com base na avaliação positiva recebida, é evidente que os utilizadores reconhecem a importância dos dados apresentados na barra inferior ao mover o rato sobre o *wafermap*, considerando-os um recurso valioso para compreender e aprofundar a análise dos resultados obtidos.

- Pergunta 19:

  No que diz respeito à pergunta de resposta aberta, obteve-se o seguinte comentário: "Foi desenvolvida uma boa aplicação para a análise dos dados paramétricos". Embora a amostra de respostas seja limitada, o facto de pelo menos um inquirido ter expressado uma opinião positiva sobre a aplicação sugere que o desenvolvimento da aplicação para análise de dados paramétricos foi bem-sucedido, fornecendo uma solução adequada às necessidades dos utilizadores.

Em resumo, a aplicação foi bem desenvolvida, pois obteve pontuações positivas na maioria das perguntas avaliadas pelos utilizadores. Isso indica que os utilizadores

consideraram a aplicação intuitiva, útil e eficiente em várias funcionalidades, como a escolha de ficheiros Excel, a personalização dos valores dos percentis, a filtragem de testes, a possibilidade de gerar *wafermaps* e a apresentação visual dos dados. As pontuações positivas refletem o cuidado em desenvolver uma aplicação que atenda às necessidades dos utilizadores e proporcione uma experiência satisfatória na análise dos dados paramétricos. Embora uma das perguntas tenha obtido uma pontuação relativamente mais baixa, indicando que a funcionalidade de guardar o estudo em formato PDF não foi considerada tão relevante, é importante relembrar que essa funcionalidade foi uma implementação adicional e não inicialmente definida nos requisitos da empresa. Portanto, apesar da pontuação inferior nessa pergunta, as avaliações positivas nas outras áreas da aplicação demonstram que a aplicação foi bem desenvolvida, atendendo em grande parte às expectativas e necessidades dos utilizadores.

Capítulo 5

Conclusões

A necessidade de produção de semicondutores tem vindo a crescer a um ritmo cada vez mais acentuado. Esta necessidade de produção deve-se ao facto de serem essenciais para a produção de uma enorme variedade de dispositivos eletrónicos presentes no nosso dia-a-dia.

Os semicondutores têm uma grande importância na produção de dispositivos eletrónicos o que obriga a existência de uma alta precisão e um grande controlo de qualidade na produção de *wafers*, principalmente quando são direcionadas para a área automóvel, uma vez que são sistemas críticos de utilização pelo Homem. Embora as *wafers* sejam uma parte fundamental da indústria moderna, elas também apresentam desafios significativos em termos de custo e eficiência. Os custos de produção são altos, e pequenas falhas no processo de fabricação podem levar a grandes prejuízos financeiros. No entanto, a procura por dispositivos eletrónicos de alta qualidade continua a crescer, impulsionando a necessidade de *wafers* com cada vez mais qualidade de fabrico e menos desperdício.

Os objetivos definidos para o desenvolvimento da aplicação foram alcançados. Foi desenvolvida uma interface gráfica do utilizador que permite a seleção de parâmetros de teste e a apresentação dos dados em formato de *wafermap* com um gradiente de cores. Adicionalmente, foram implementadas mais funcionalidades, como a escolha dos valores dos percentis, barra superior e inferior com funcionalidades suplementares e a possibilidade de guardar as informações analisadas em formato PDF. A aplicação contém todos os objetivos definidos, fornecendo uma solução eficiente e intuitiva para a visualização e análise de resultados paramétricos de *wafers*.

Durante o processo de desenvolvimento da aplicação, encontraram-se algumas dificuldades na implementação da introdução e leitura do ficheiro, bem como no cálculo dos percentis para obter os intervalos de valores associados a cada cor. Além disso, enfrentou-se a dificuldade da representação visual da *wafer* que se adaptasse dinamicamente ao número de *dies* existentes e também na informação presente na barra inferior relativa às coordenadas dos diversos *dies*.

A implementação da introdução e leitura do ficheiro exigiu um estudo cuidadoso sobre a estrutura e o formato dos dados, assegurando que todas as informações relevantes fossem devidamente processadas e armazenadas. No que diz respeito ao cálculo dos percentis, foi necessária uma análise estatística dos dados, assim como a seleção das métricas adequadas para o seu cálculo.

No entanto, a maior dificuldade encontrada foi criar uma representação dinâmica da *wafer*, capaz de se ajustar automaticamente ao número variável de *dies* presentes. Além disso, houve a necessidade de compreender os princípios de interação entre o rato e o *TabControl*, para exibir corretamente as coordenadas dos dies na barra inferior.

As dificuldades encontradas no desenvolvimento destas funcionalidades foram superadas através de uma pesquisa detalhada, experimentação e iterações no processo de desenvolvimento. Compreender o funcionamento dos problemas e aplicar soluções adequadas permitiu alcançar os objetivos propostos para a aplicação.

De uma forma geral, o conhecimento obtido através da elaboração de um projeto desta magnitude que introduziu tantos conceitos, alguns deles novos, permitiu entender melhor as etapas no processo de fabrico de componentes eletrónicos e a importância que tem o controlo de qualidade na sua produção.

Em suma, foi possível consolidar e adquirir diversos conhecimentos não só a nível de *software*, mas também relacionados com o mundo dos semicondutores. As dificuldades que foram aparecendo ao longo do projeto derivaram da falta de conhecimento em linguagens de programação até então desconhecidas. A elaboração deste projeto ajudou não só a aumentar, como também a consolidar os conhecimentos adquiridos ao longo de todo o percurso académico. Tendo a elaboração deste projeto um ponto de vista mais prático, integrado num estágio em empresa, deu uma perspetiva diferente e fez com seja uma mais-valia para o mundo profissional.

5.1 Trabalho Futuro

Embora os principais objetivos tenham sido cumpridos com sucesso, existem sempre melhorias que podem ser feitas, principalmente quando se trata de uma aplicação que iniciou o seu desenvolvimento de novo.

Começando pelas *listbox*, em vez de as usar elementos separados para apresentar os nomes e números dos testes, seria uma melhoria substituí-las por uma *listview*

compacta. Essa mudança proporcionaria uma experiência mais amigável para o utilizador. Em relação à leitura do ficheiro Excel, em vez de carregar todos os dados de uma só vez para memória, seria mais eficiente carregar apenas o cabeçalho, que contém os nomes e números dos testes. Deste modo, só depois, à medida que o utilizador seleciona um teste específico, apenas as informações necessárias para esse teste seriam carregadas. Por fim, uma última melhoria seria adicionar uma opção para realizar a rotação no plano da *wafer*, assim como a possibilidade de fazer zoom na sua visualização.

Estas melhorias, visariam proporcionar uma experiência mais intuitiva, eficiente e flexível aos utilizadores, aumentando a experiência do utilizador e também a funcionalidade global do sistema.

Referências

[1] Amkor, "Careers – portugal." (Último Acesso em 17/03/2023). [Citado nas páginas vii e 3]

[2] L. L. e Andrzej Jakubowski, "History of semiconductors," *Journal of Telecommunications and Information Technology*, 2010. [Citado na página 7]

[3] D. Klein, "The history of semiconductor memory: From magnetic tape to nand flash memory," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 16–22, 2016. [Citado na página 7]

[4] M. Vasilevskiy, "Um semicondutor vale mais do que um condutor?," *Correio Do Minho*, 2014. [Citado nas páginas 7 e 8]

[5] Amkor, "Wafer level test overview," 2017. [Citado nas páginas vii, 8 e 9]

[6] WaferPro, "What is a semiconductor wafer?." (Último Acesso em 19/03/2023). [Citado nas páginas vii, 8, 9 e 10]

[7] A. F. A. de Matos, "Melhoria do controlo dos materiais na indústria de semicondutores: caso de estudo," Master's thesis, ISEP, Porto, 2019. [Citado nas páginas vii, 11, 12 e 13]

[8] AnySilicon, "Understanding wafer level packing." (Último Acesso em 29/03/2023). [Citado nas páginas vii e 12]

[9] JavaTPoint, "Programming language." (Último Acesso em 20/07/2023). [Citado nas páginas vii, 13, 15 e 16]

[10] C. Team, "What is a programming language?." (Último Acesso em 30/07/2023). [Citado na página 13]

[11] R. Sheldon, "C# (c-sharp)." (Último Acesso em 30/07/2023). [Citado na página 14]

[12] Microsoft, "A tour of the c# language." (Último Acesso em 30/07/2023). [Citado na página 14]

[13] X. Baptista, "As vantagens e desvantagens do c# (c-sharp)." (Último Acesso em 30/07/2023). [Citado na página 14]

[14] JavaTPoint, “Role of python in artificial intelligence.” (Último Acesso em 30/07/2023). [Citado na página 15]

[15] A. Subasi, “Pratical machine learning for data analysis using python.” (Último Acesso em 30/07/2023). [Citado na página 15]

[16] DataFlair, “Advantages and disadvantages of python – how it is dominating programming world.” (Último Acesso em 30/07/2023). [Citado nas páginas 15 e 16]

[17] IBM, “What is java?.” (Último Acesso em 30/07/2023). [Citado na página 16]

[18] M. Tyson, “What is the jvm? introducing the java virtual machine.” (Último Acesso em 30/07/2023). [Citado na página 16]

[19] P. Morais, “Vantagens e desvantagens da linguagem java.” (Último Acesso em 30/07/2023). [Citado na página 17]

[20] falticon, “Java icon.” (Último Acesso em 30/07/2023). [Citado nas páginas vii e 17]

[21] Microsoft, “.net desktop guide for windows forms.” (Último Acesso em 19/03/2023). [Citado na página 18]

[22] Microsoft, “Desktop guide (windows forms .net).” (Último Acesso em 22/03/2023). [Citado na página 19]

[23] M. Project, “Windows forms c - forms and dialogs.” (Último Acesso em 30/03/2023). [Citado nas páginas vii e 19]

[24] Microsoft, “Guia da área de trabalho (wpf .net).” (Último Acesso em 22/03/2023). [Citado na página 20]

[25] DEVMEDIA, “Windows forms x wpf.” (Último Acesso em 22/03/2023). [Citado na página 20]

[26] V. Kaplan, “Using a scripting language to develop native windows wpf gui apps.” (Último Acesso em 30/03/2023). [Citado nas páginas vii e 20]

[27] Qt, “About qt.” (Último Acesso em 24/03/2023). [Citado nas páginas vii, 21 e 22]

[28] A. Solovev, “Qt for embedded development: The many pros and the few cons.” (Último Acesso em 24/03/2023). [Citado na página 21]

[29] Qt, “Qt quick controls 1 - ui forms.” (Último Acesso em 30/03/2023). [Citado nas páginas vii e 21]

[30] GTK, “Gtk.” (Último Acesso em 30/07/2023). [Citado na página 22]

[31] GTK, “Gtk – 3.0.” (Último Acesso em 24/03/2023). [Citado na página 22]

[32] Terrance, "Where are my gtk examples?." (Último Acesso em 30/03/2023). [Citado nas páginas vii e 23]

[33] C. Wille, "Sharpdevelop wiki." (Último Acesso em 30/07/2023). [Citado na página 24]

[34] C. L. Sharpe, "Why is sharpdevelop ide so less known and unpopular considering its lightweight, open-source and capable of most things possibe in .net?." (Último Acesso em 30/07/2023). [Citado na página 24]

[35] uptodown, "Sharpdevelop." (Último Acesso em 30/07/2023). [Citado nas páginas vii e 24]

[36] Microsoft, "What is visual studio?." (Último Acesso em 30/07/2023). [Citado na página 25]

[37] W. Commons, "File:visual studio icon 2019.svg." (Último Acesso em 30/07/2023). [Citado nas páginas vii e 25]

Anexo A

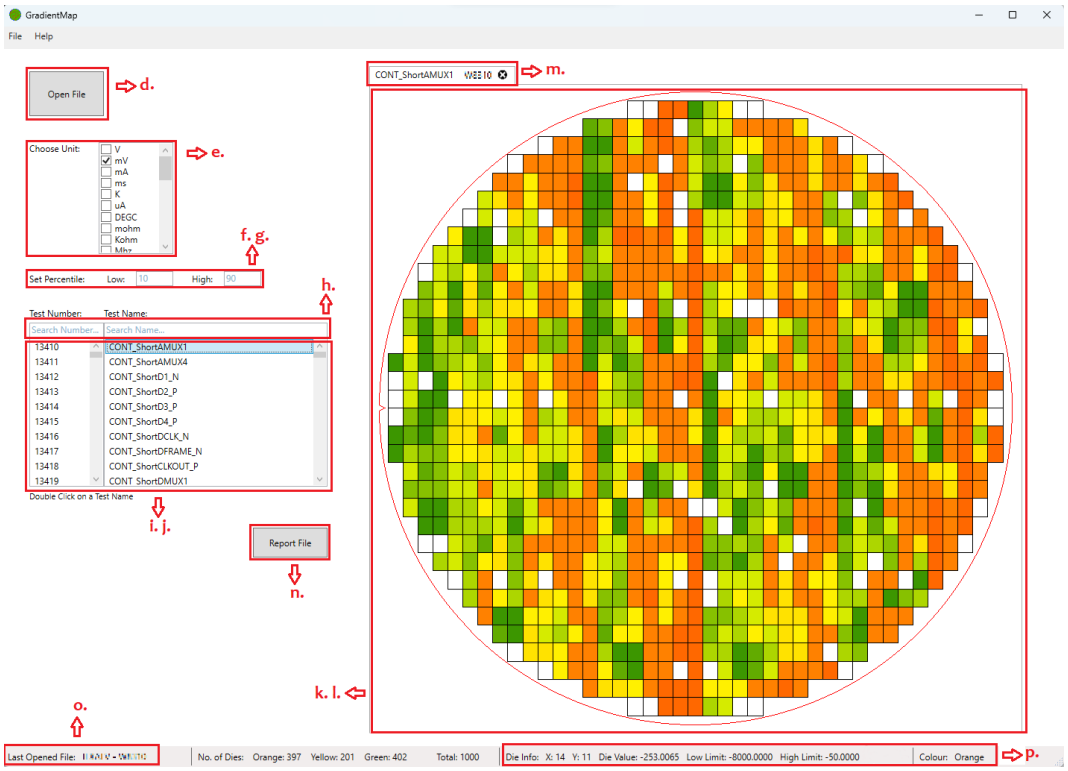
Inquérito

Inquérito de satisfação relativo à aplicação de apresentação de dados paramétricos no formato de wafer *map*

O presente inquérito está integrado no desenvolvimento do trabalho de tese/dissertação do Mestrado em Automação e Sistemas do Instituto Superior de Engenharia do Porto, visando avaliar a aplicação desenvolvida para a empresa Amkor Technology e verificar se os objetivos propostos foram alcançados.

Desde já é garantido a total confidencialidade e anonimato dos dados recolhidos, sendo os mesmos utilizados unicamente para a realização deste estudo. Os resultados serão utilizados para validação do trabalho, sendo apresentados no relatório final da tese/dissertação.

Obrigado pela sua disponibilidade e colaboração.


GradientMap
File
Help
Open File
d.
Choose Unit:
V
mV
mA
ms
K
uA
DEGC
mohm
Kohm
e.
f. g.
Set Percentile:
Low:
10
High:
90
h.
Test Number:
Test Name:
Search Number...
Search Name...
13410 CONT_ShortAMUX1
13411 CONT_ShortAMUX4
13412 CONT_ShortD1_N
13413 CONT_ShortD2_P
13414 CONT_ShortD3_P
13415 CONT_ShortD4_P
13416 CONT_ShortDCLK_N
13417 CONT_ShortDFRAME_N
13418 CONT_ShortCLKOUT_P
13419 CONT_ShortDMUX1
Double Click on a Test Name
i. j.
Report File
n.
CONT_ShortAMUX1
m.
k. l.
o.
Last Opened File:
No. of Dies: Orange: 397 Yellow: 201 Green: 402 Total: 1000
Die Info: X: 14 Y: 11 Die Value: -253.0065 Low Limit: -8000.0000 High Limit: -50.0000
Colour: Orange
p.

a. Como avalia o aspeto (*layout*) da interface *

	1	2	3	4	5	
Muito Mau	○	○	○	○	○	Muito Bom

b. Como avalia a facilidade de utilização (interação com os menus) *

	1	2	3	4	5	
Muito Difícil	○	○	○	○	○	Muito Fácil

c. Como avalia os tempos de resposta (desempenho) da aplicação *

	1	2	3	4	5	
Muito Maus	○	○	○	○	○	Muito Bons

d. Avalie a facilidade de escolha de um ficheiro Excel de um teste paramétrico *

	1	2	3	4	5	
Muito Dificil	○	○	○	○	○	Muito Fácil

e. Avalie de que forma a escolha das unidades da lista apresentada é adequada *

	1	2	3	4	5	
Pouco Adequada	○	○	○	○	○	Muito Adequada

f. Como avalia a facilidade de perceção dos valores de *default* dos percentis. *

	1	2	3	4	5	
Muito Difícil	○	○	○	○	○	Muito Fácil

g. Como avalia a possibilidade de alterar os valores dos percentis *

	1	2	3	4	5	
Muito Mau	○	○	○	○	○	Muito Bom

h. Como avalia a utilidade das caixas de pesquisa (*search bar*), *Test Number* e *Test Name* *

	1	2	3	4	5	
Nada útil	○	○	○	○	○	Muito útil

i. Como avalia a apresentação dos testes disponíveis, na listagem interativa lateral (*Test Number* e *Test Name*) *

	1	2	3	4	5	
Muito Má	○	○	○	○	○	Muito Boa

j. Como avalia a facilidade de gerar um *wafermap* *

	1	2	3	4	5	
Muito Difícil	○	○	○	○	○	Muito Fácil

k. Avalie a visualização dos dados sob a forma de *wafermap* *

	1	2	3	4	5	
Muito Má	○	○	○	○	○	Muito Boa

l. Avalie de que forma o gradiente de cor é apresentado de forma clara e adequada. *

	1	2	3	4	5	
Muito Mau	○	○	○	○	○	Muito Bom

m. Como avalia a importância da possibilidade de abrir vários separadores (*wafermap*) correspondentes a diferentes testes *

	1	2	3	4	5	
Pouco Importante	○	○	○	○	○	Muito Importante

n. Como avalia a possibilidade de poder guardar em **pdf** o estudo sobre um teste (*wafermap*) *

	1	2	3	4	5	
Pouco útil	○	○	○	○	○	Muito útil

o. Como avalia a importância da informação relativa ao último ficheiro aberto, presente no lado esquerdo da barra inferior *

	1	2	3	4	5	
Pouco Importante	○	○	○	○	○	Muito Importante

p. Como avalia a importância dos dados obtidos na barra inferior quando se move o rato por cima do *wafermap* (coordenadas, valor do teste e cor do *die*) *

	1	2	3	4	5	
Pouco Importante	○	○	○	○	○	Muito Importante

q. Como avalia a facilidade em encontrar as informações pretendidas, para uma análise de um teste *

	1	2	3	4	5	
Muito Difícil	○	○	○	○	○	Muito Fácil

r. Classifique a adequação das funcionalidades desenvolvidas, face aos objetivos iniciais do utilizador *

	1	2	3	4	5	
Muito Más	○	○	○	○	○	Muito Boas

s. Classifique de que forma a aplicação melhorou a análise dos dados paramétricos de testes *

	1	2	3	4	5	
Não melhorou	○	○	○	○	○	Melhorou muito

t. De forma geral como classifica a aplicação desenvolvida? *

	1	2	3	4	5	
Muito Má	○	○	○	○	○	Muito Boa

u. Existe algum aspeto específico que gostaria de mencionar em relação à aplicação desenvolvida?

A sua resposta

Observações adicionais

A sua resposta