Universidade Tuiuti do Paraná
Curso: Ciência da Computação

1º Estudo Dirigido de Sistemas Operacionais - Prof. Baroni

Alunos: Gabriel Bauer / Iuker de Souza Santos

Projeto ”Meu Linux, Minha Vida”

Criação de Distribuição Linux Personalizada

Sumário

Parte I

Proposta de Projeto

1 Visão Geral da Distribuição

- **Nome:** iubaOS
- **Identidade Conceitual:** Foco no meio acadêmico, combinando leveza e funcionalidade.

1.1 Público-alvo e Casos de Uso Pretendidos

A iubaOS é projetada para atender diretamente às necessidades acadêmicas de Estudantes Universitários e Professores, integrando ferramentas essenciais ao seu fluxo de trabalho diário. Os principais casos de uso que a iubaOS visa simplificar incluem:

- **Acesso e Gestão de Notas e Informações Acadêmicas:** Professores poderão lançar notas e visualizar médias, enquanto estudantes terão acesso facilitado para consultar seu desempenho através do aplicativo **Notesyst**.
- **Organização e Acompanhamento de Tarefas:** Um aplicativo de ”To-Do List” nativo permitirá que usuários listem e gerenciem suas atividades pendentes, auxiliando na organização do tempo.

1.2 Filosofia e Princípios Orientadores

A iubaOS acredita que a tecnologia deve ser uma ferramenta poderosa e acessível para o aprendizado e o ensino. Sua filosofia é criar um ambiente operacional dedicado às necessidades do universo acadêmico.

- **Foco Inabalável no Usuário Acadêmico:** Todas as decisões de design foram tomadas considerando a experiência de estudo, pesquisa e ensino no nível universitário.
- **Ferramentas Integradas e Relevantes:** Priorizar a inclusão de software essencial e de alta qualidade para fins acadêmicos.
- **Estabilidade e Confiabilidade:** Prover uma plataforma robusta e estável para as atividades acadêmicas, baseada em uma versão de Suporte de Longo Prazo (LTS).
- **Segurança e Privacidade dos Dados:** Implementar configurações e ferramentas que protejam as informações pessoais e acadêmicas dos usuários.

2 Especificações Técnicas

- **Base da distribuição:** Xubuntu 22.04 LTS.
- **Arquitetura(s) de hardware suportada(s):** amd64 (ou x86_64), para processadores Intel e AMD de 64 bits.

2.0.1 Requisitos mínimos de sistema

- **Processador:** Intel Core i3 de 2ª Geração ou AMD Ryzen 3 da 1ª Geração.
- **Memória RAM:** 4GB.
- **Espaço mínimo de disco:** 20GB.

2.1 Componentes Fundamentais

- **Versão do Kernel:** Baseado no Kernel LTS do Ubuntu 22.04.
- **Sistema de inicialização:** systemd.
- **Sistema de gerenciamento de pacotes:** APT (Advanced Package Tool), com suporte a pacotes .deb.

3 Design de Sistema

- **Ambiente desktop/interface de usuário:** XFCE, conhecido por seu baixo consumo de recursos, estabilidade e alta capacidade de personalização.

3.0.1 Aplicativos padrão a serem incluídos

- **NoteSyst:** Aplicativo personalizado para gestão de notas acadêmicas.
- **To-Do List:** Aplicativo personalizado para organização de tarefas.
- **Batalha Naval:** Jogo personalizado para entretenimento.
- **Navegadores Web:** Firefox (padrão) e Google Chrome (adicional).
- **Desenvolvimento:** Visual Studio Code.
- **Utilitários Essenciais:** LibreOffice, Gerenciador de Arquivos (Thunar), Calculadora e Terminal.

3.0.2 Personalizações específicas de sistema

- **Tema:** Orchis (variante Grey-Dark), para uma aparência moderna e escura.
- **Ícones:** Papirus (variante Dark), um conjunto de ícones completo e visualmente coeso.
- **Fontes:** Padrão do sistema (Noto Sans).
- **Papel de Parede:** Imagem de fundo personalizada.
- **Configurações de janelas:** Botões e comportamento ajustados pelo tema Orchis.

4 Plano de Implementação

4.0.1 Ferramentas de Desenvolvimento e Teste

- **Criação da Imagem ISO:** Cubic (Custom Ubuntu ISO Creator).
- **Customização:** Ambiente chroot (fornecido pelo Cubic), apt/dpkg, Shell Scripting e editores de texto (nano).
- **Teste Primário:** VirtualBox em um sistema Linux host para testes rápidos e seguros.
- **Teste Secundário:** Dual-boot em hardware real para verificação de compatibilidade.

4.0.2 Cronograma de Marcos

O planejamento inicial serviu como base, mas a execução real exigiu ajustes na ordem e na complexidade das tarefas.

Fase 1: Configuração do Ambiente Preparar o sistema host com as ferramentas necessárias (Cubic, VirtualBox).

Fase 2: Construção da Base Mínima Utilizar a ISO do Xubuntu como ponto de partida dentro do Cubic.

Fase 3: Integração do Ambiente de Desktop Aplicar as personalizações visuais, incluindo a instalação e configuração padrão do tema Orchis e dos ícones Papirus no ambiente XFCE.

Fase 4: Implementação dos Aplicativos Integrar os aplicativos Python personalizados, softwares de terceiros (Chrome, VS Code) e remover pacotes indesejados, além de resolver todas as dependências de software.

Nota: A execução real demonstrou que as Fases 3 e 4 ocorreram de forma mais entrelaçada do que o planejado inicialmente.

5 Potenciais Desafios e Estratégias

- **Desafio:** Aprender a usar as ferramentas de construção (Cubic).
  - *Estratégia:* Seguir um processo iterativo de tentativa e erro, começando com modificações simples e aumentando a complexidade gradualmente.
- **Desafio:** Conflitos de dependência entre pacotes.
  - *Estratégia:* Priorizar pacotes dos repositórios oficiais do Ubuntu base e utilizar 'pip' como alternativa para bibliotecas Python não disponíveis, resolvendo o problema do 'tkcalendar' e do 'customtkinter'.
- **Desafio:** O ambiente de desktop não aplicar as personalizações corretamente.
  - *Estratégia:* Utilizar a pasta '/etc/skel' para criar arquivos de configuração padrão para novos usuários, garantindo que temas e ícones sejam aplicados na primeira inicialização.
- **Desafio:** Manter o tamanho da ISO gerenciável.

- *Estratégia:* Incluir apenas o software necessário, realizar a remoção de pacotes padrão com ‘apt purge‘ e limpar o cache do APT e os arquivos temporários de instalação ao final do processo.

6 Análise Conceitual

O projeto iubaOS explorará e implementará os conceitos fundamentais de sistemas operacionais. A implementação prática desses conceitos acontecerá através da configuração, seleção de ferramentas e estruturação do sistema final. Os principais conceitos abordados são:

- **Gerenciamento de Processos:** Tratado pelo Kernel Linux, que agenda tarefas na CPU. A distro fornecerá ferramentas de monitoramento como ‘top‘ e ‘htop‘.
- **Gerenciamento de Memória:** Feito pelo Kernel (RAM e swap). A distro incluirá ferramentas de monitoramento.
- **Sistema de Arquivos:** O padrão será o Ext4, com a estrutura FHS (Filesystem Hierarchy Standard), que foi respeitada ao colocar os aplicativos (‘/opt‘, ‘/usr/local/bin‘) e temas (‘/usr/share/themes‘, ‘/usr/share/icons‘).
- **Segurança:** Implementada através do modelo de permissões do Linux, autenticação (PAM), firewall (ufw), e o sistema de atualizações (APT), que foi mantido funcional para os aplicativos de terceiros.

6.1 Diagrama Conceitual da Estrutura da iubaOS

O diagrama abaixo ilustra as camadas da distribuição, desde o hardware até a interação com o usuário final.

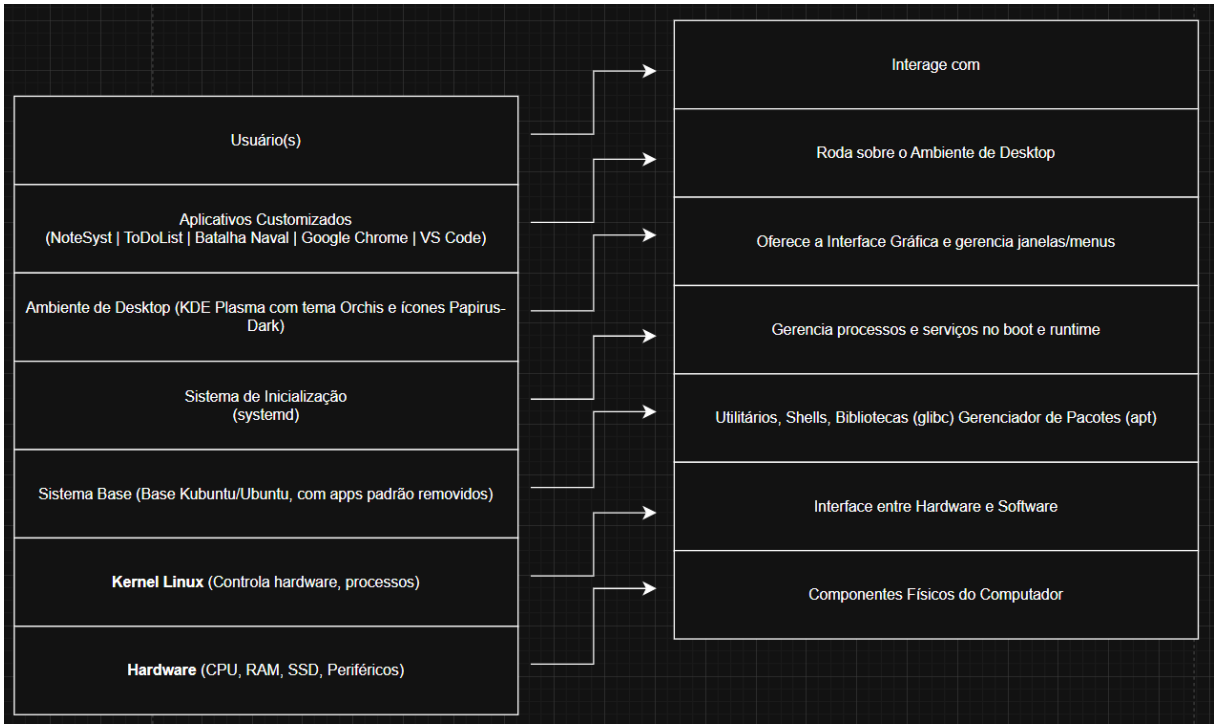

Figura 1: Diagrama Conceitual da Estrutura da iubaOS.

Parte II

Execução e Implementação

7 Relato da Execução Prática

O processo de execução do projeto, batizado de iubaOS, foi uma jornada iterativa que, embora tenha atingido os objetivos propostos, divergiu em vários pontos do planejamento inicial. A intenção primária era a criação de uma distribuição Linux com um conjunto específico de aplicações Python personalizadas. No entanto, o escopo foi significativamente expandido para incluir a instalação de softwares de terceiros (Google Chrome, VS Code), a remoção de aplicativos padrão e uma profunda personalização da interface gráfica.

A escolha da distribuição base foi um dos principais pontos de evolução. Embora outras bases como Xubuntu e Linux Mint tenham sido consideradas, a decisão final recaiu sobre o **Xubuntu**, utilizando o ambiente de desktop KDE Plasma, que ofereceu um balanço desejado entre recursos e estética. Os desafios técnicos encontrados foram majoritariamente relacionados à integração de componentes não-nativos (scripts Python, temas) e à configuração de padrões de sistema, exigindo uma abordagem de depuração e pesquisa mais aprofundada do que o previsto.

Registro de Decisões Técnicas e Suas Justificativas

Durante a implementação, diversos desafios técnicos surgiram, necessitando de decisões específicas para garantir a estabilidade e a funcionalidade da distribuição final.

- **Configuração do Papel de Parede Padrão:** O desafio inicial foi definir um método que aplicasse um papel de parede de forma consistente no primeiro boot, especialmente em um ambiente *live*. A pesquisa revelou que cada ambiente de desktop (XFCE, Cinnamon, KDE Plasma) possui um mecanismo distinto. Para o KDE Plasma, a solução mais robusta foi a criação de um script de inicialização automática ('.desktop' em '/etc/skel/.config/autostart/'). Este script utiliza o comando nativo 'plasma-apply-wallpaperimage' após um breve atraso ('sleep 5'), garantindo que a área de trabalho esteja totalmente carregada antes de aplicar a imagem. Esta abordagem se mostrou mais eficaz do que apenas modificar arquivos de configuração estáticos, que nem sempre eram aplicados no modo *live*.
- **Execução dos Scripts Python Personalizados:** Este foi o desafio técnico mais complexo. Os aplicativos, embora funcionalmente corretos, não executavam ao serem clicados. A depuração, realizada ao tentar executá-los via terminal, revelou duas causas raiz distintas:
    1. **Formato de Quebra de Linha (CRLF vs. LF):** O primeiro erro diagnosticado foi "/usr/bin/env: 'python3˚: No such file or directory". Isso indicou que os arquivos '.py' foram salvos com terminações de linha do Windows (CRLF). O sistema Linux interpretava o caractere invisível de retorno de carro ('̊) como parte do nome do interpretador. A decisão técnica foi instalar a ferramenta 'dos2unix' dentro do ambiente Cubic e utilizá-la para converter todos os scripts para o formato Unix (LF) antes de copiá-los para suas pastas finais no sistema.
    2. **Ausência do Shebang:** Um erro anterior, 'import: command not found', revelou que os scripts não estavam sendo interpretados como arquivos Python. A solução foi a padronização de todos os scripts executáveis com a linha shebang '!/usr/bin/env python3' no topo, instruindo o sistema operacional a usar o interpretador Python 3 para executá-los.
- **Gerenciamento de Dependências e Temas:** A instalação de pacotes de fora dos repositórios padrão apresentou desafios. A biblioteca Python 'customtkinter' precisou ser instalada via 'pip'. Isso, por sua vez, gerou o erro 'externally-managed-environment' (PEP 668) nas versões recentes do Xubuntu. A decisão foi usar a flag '–break-system-packages', uma medida justificada no contexto de construção de uma imagem de sistema controlada, onde o administrador (nós) assume a responsabilidade pela integridade do ambiente Python global. Similarmente, o tema de ícones Papirus foi instalado via 'apt', enquanto o tema Orchis foi instalado via 'git clone' e seu script de instalação, demonstrando uma abordagem flexível para a obtenção de pacotes.

Diário de Desenvolvimento

O desenvolvimento do iubaOS ocorreu ao longo de aproximadamente um mês, segue abaixo o desenvolvimento:

- **Semana 1 (19/05 a 25/05):** Foco total no desenvolvimento dos três aplicativos principais em Python. O "Batalha Naval" foi desenvolvido utilizando Pygame e CustomTkinter para a interface do menu. O "Notesyst" e o "Todolist" foram criados utilizando CustomTkinter,

Tkinter e bibliotecas como Matplotlib e TkCalendar, estabelecendo a base de software personalizado da distro.

- **Semana 2 (26/05 a 01/06):** Início da fase de integração com a ferramenta Cubic. A primeira versão da ISO foi montada, e os aplicativos Python foram adicionados ao sistema. Nesta fase, surgiram os primeiros e mais críticos desafios de execução. Grande parte da semana foi dedicada à depuração via terminal para diagnosticar e resolver os problemas de *shebang* ausente e, posteriormente, o sutil erro de quebra de linha do Windows (‘̊).
- **Semana 3 (02/06 a 08/06):** Com os programas personalizados funcionando, o escopo do projeto foi expandido. Foram adicionados ao processo de construção o Google Chrome e o VS Code, baixados como pacotes ‘.deb‘. Iniciou-se o processo de ”limpeza” do sistema, com a remoção de aplicativos padrão do Xubuntu via ‘apt purge‘. A instalação de dependências Python também foi finalizada, superando os desafios do ‘pip‘ e do ‘externally-managed-environment‘.
- **Semana 4 (09/06 a 19/06):** O foco desta última semana foi a personalização visual e a finalização. Foram pesquisados, baixados e instalados o tema Orchis GTK/Plasma e o tema de ícones Papirus. O principal desafio foi garantir que a tematização ”dark/black” fosse aplicada de forma consistente e automática. Isso foi resolvido com a criação de múltiplos arquivos de configuração no diretório ‘/etc/skel‘. Ocorreram os últimos ajustes, como a correção de permissões de arquivos no diretório do projeto que impediam a geração final da ISO. No dia 19 de junho, o script final foi consolidado e a versão candidata a final do iubaOS foi gerada com sucesso.

8 Análise Detalhada dos Scripts de Personalização

A seguir, apresentamos e analisamos os scripts utilizados para configurar a distribuição iubaOS. Cada bloco de código é acompanhado de uma explicação detalhada sobre sua função dentro do ambiente de construção do CUBIC.

8.1 Instalação dos Programas (Batalha Naval, NotesSyst, ToDoList)

Esta seção do script é responsável por instalar e integrar completamente os três programas Python ao sistema, incluindo dependências, ícones e atalhos.

8.1.1 Verificação e Criação de Diretórios Iniciais

Estes comandos são usados para preparar o ambiente dentro do CUBIC. Primeiro, ele lista (‘ls‘) o conteúdo da pasta ‘/root/‘. Em seguida, cria (‘mkdir‘) um diretório chamado ‘meus-programas‘ para organizar os arquivos dos projetos. Por fim, outros comandos ‘ls‘ e ‘tree‘ servem para verificar se o diretório foi criado corretamente.

```
ls /root/
mkdir /root/meus-programas
ls /root/
ls /root/meus-programas/
tree /root/meus-programas/
```

Listing 1: Preparação do ambiente no CUBIC

8.1.2 Instalação de Dependências Python

Este bloco instala todas as bibliotecas que os programas Python precisam para funcionar. Primeiro, o ‘apt update‘ atualiza a lista de pacotes disponíveis. Depois, o ‘apt install -y‘ instala bibliotecas do sistema, como Pygame e Tkinter. Por fim, o ‘pip install‘ busca pacotes que não estão nos repositórios do sistema, como o ‘customtkinter‘, diretamente do Python Package Index (PyPI).

```
apt update && apt install -y python3-pygame python3-tk python3-pil.imagetk
    python3-matplotlib python3-pip
pip install customtkinter tkcalendar
```

Listing 2: Instalação de dependências Python via APT e PIP

8.1.3 Criação dos Diretórios de Destino

Este comando cria as pastas onde os programas ficarão instalados permanentemente. O diretório ‘/opt‘ é o local padrão no Linux para softwares opcionais ou de terceiros. A opção ‘-p‘ garante que os diretórios pais sejam criados caso não existam.

```
mkdir -p /opt/batalha-naval
mkdir -p /opt/notesyst
mkdir -p /opt/todolist
mkdir -p /usr/share/pixmaps
```

Listing 3: Criação de diretórios em /opt e /usr/share/pixmaps

8.1.4 Cópia dos Arquivos dos Programas

Copia os arquivos dos projetos da pasta temporária ‘/root/meus-programas/‘ para os diretórios permanentes em ‘/opt/‘. A opção ‘-r‘ significa ”recursivo”, garantindo que todas as subpastas e arquivos sejam copiados.

```
cp -r /root/meus-programas/batalha-naval/* /opt/batalha-naval/
cp -r /root/meus-programas/notesyst/* /opt/notesyst/
cp -r /root/meus-programas/todolist/* /opt/todolist/
```

Listing 4: Copiando arquivos dos projetos para /opt

8.1.5 Criação de Lançadores Globais

Cria pequenos scripts em ‘/usr/local/bin/‘ que permitem executar os programas pelo nome de qualquer lugar no terminal. Cada script entra na pasta do programa (‘cd‘) e o executa com o Python (‘python3‘).

```
echo '#!/bin/bash
cd /opt/batalha-naval/
python3 main.py' > /usr/local/bin/batalha-naval

echo '#!/bin/bash
cd /opt/notesyst/
python3 universidade.py' > /usr/local/bin/sistema-notas

echo '#!/bin/bash
cd /opt/todolist/
python3 todo.py' > /usr/local/bin/todolist
```

Listing 5: Scripts para lançar os aplicativos pelo terminal

8.1.6 Criação de Atalhos no Menu de Aplicativos (.desktop)

Cria os arquivos '.desktop', que são os "atalhos" que aparecem no menu de aplicativos do sistema. Cada arquivo define o nome, a descrição, o comando a ser executado ('Exec=') e o ícone de um programa.

```
echo '[Desktop Entry]
Version=1.0
Name=Batalha Naval
...
Categories=Game;' > /usr/share/applications/batalha-naval.desktop

echo '[Desktop Entry]
Version=1.0
Name=NotesSyst
...
Categories=Office;Education;' > /usr/share/applications/notesyst.desktop

echo '[Desktop Entry]
Version=1.0
Name=ToDoList
...
Categories=Office;Utility;' > /usr/share/applications/todolist.desktop
```

Listing 6: Criação dos arquivos .desktop para o menu

8.1.7 Cópia dos Ícones e Aplicação de Permissões

Copia os ícones ('.png') para '/usr/share/pixmaps/', um diretório padrão onde o sistema os procura. Em seguida, o 'chmod' ajusta as permissões: '755' para os lançadores (para serem executáveis) e '644' para os ícones e atalhos (apenas leitura para outros usuários).

```
# Copia dos  cones
cp /root/meus-programas/batalha-naval/navio.png /usr/share/pixmaps/batalha-naval
    .png
cp /root/meus-programas/notesyst/2.png /usr/share/pixmaps/notesyst.png
cp /root/meus-programas/todolist/5.png /usr/share/pixmaps/todolist.png

# Permiss es
chmod 755 /usr/local/bin/batalha-naval
chmod 755 /usr/local/bin/sistema-notas
chmod 755 /usr/local/bin/todolist
chmod 644 /usr/share/pixmaps/*.png
chmod 644 /usr/share/applications/*.desktop
chmod -R 755 /opt/*
```

Listing 7: Cópia de ícones e aplicação de permissões

8.1.8 Atualização da Base de Dados e Atalhos na Área de Trabalho

O comando 'update-desktop-database' força o sistema a reconhecer os novos atalhos no menu. Depois, os atalhos são copiados para '/etc/skel/Desktop', garantindo que todo novo usuário criado no sistema já comece com eles em sua Área de Trabalho.

```
update-desktop-database /usr/share/applications/

mkdir -p /etc/skel/Desktop
cp /usr/share/applications/batalha-naval.desktop /etc/skel/Desktop/
cp /usr/share/applications/notesyst.desktop /etc/skel/Desktop/
```

```
cp /usr/share/applications/todolist.desktop /etc/skel/Desktop/
chmod +x /etc/skel/Desktop/*.desktop
```

Listing 8: Finalizando a integração com o sistema

8.2 Instalação de Software Adicional

Esta seção do script cuida da instalação de dois programas populares de fontes externas.

8.2.1 Instalação do Google Chrome

O comando 'wget' baixa o arquivo instalador oficial do Google Chrome ('.deb'). Em seguida, 'apt-get install -y ./' instala o pacote localmente, sem procurá-lo nos repositórios online.

```
wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
sudo apt-get install -y ./google-chrome-stable_current_amd64.deb
```

Listing 9: Instalação do Google Chrome

8.2.2 Instalação do Visual Studio Code

O processo é idêntico ao do Chrome. A opção '-O' no 'wget' permite renomear o arquivo baixado, facilitando o comando de instalação.

```
wget -O vscode.deb https://go.microsoft.com/fwlink/?LinkID=760868
sudo apt-get install -y ./vscode.deb
```

Listing 10: Instalação do Visual Studio Code

8.3 Ajustes Finais do Sistema

Esta última parte do script personaliza a aparência do sistema e remove softwares indesejados.

8.3.1 Remoção de Aplicativos e Instalação de Dependências de Tema

O comando 'apt-get purge -y' remove completamente uma lista de programas pré-instalados, incluindo seus arquivos de configuração. Depois, 'apt-get install' baixa as ferramentas necessárias para o novo tema, como 'git' para baixar o tema, 'kvantum' para aplicar o tema em apps Qt e o pacote de ícones 'papirus-icon-theme'.

```
sudo apt-get purge -y thunderbird hexchat pidgin parole rhythmbox xfburn gnome-
    mines gnome-sudoku simple-scan transmission-gtk
sudo apt-get install -y git sassc libglib2.0-dev-bin qt5-style-kvantum papirus-
    icon-theme
```

Listing 11: Limpeza e instalação de dependências de tema

8.3.2 Download e Instalação do Tema

Baixa o código-fonte do tema Orchis do GitHub para a pasta '/tmp' (para arquivos temporários) e executa seu script de instalação, que instala a variante escura com detalhes em cinza.

```
cd /tmp
git clone https://github.com/vinceliuice/Orchis-theme.git
cd Orchis-theme
sudo ./install.sh -c dark -t grey --tweaks black
cd /
```

Listing 12: Instalação do Tema Orchis

8.3.3 Configuração da Aparência Padrão

Este é um passo crucial que define o novo tema e ícones como padrão para todos os novos usuários do sistema. Ele cria arquivos de configuração em '/etc/skel/' que definem o tema 'Orchis-Grey-Dark' e os ícones 'Papirus-Dark' para aplicações GTK (como Firefox) e Plasma.

```
# Garante que as pastas de configura  o existam
sudo mkdir -p /etc/skel/.config/gtk-3.0/
sudo mkdir -p /etc/skel/.config/Kvantum/
sudo mkdir -p /etc/profile.d/

# For a o uso do Kvantum
sudo cat > /etc/profile.d/qt-style.sh << EOF
export QT_STYLE_OVERRIDE=kvantum
EOF

# Configura o tema para GTK
sudo cat > /etc/skel/.config/gtk-3.0/settings.ini << EOF
[Settings]
gtk-application-prefer-dark-theme=true
gtk-theme-name=Orchis-Grey-Dark
gtk-icon-theme-name=Papirus-Dark
gtk-font-name=Noto Sans 10
EOF

# Configura o tema do Plasma
sudo cat > /etc/skel/.config/plasmarc << EOF
[Theme]
name=Orchis-Grey-Dark
EOF
```

Listing 13: Configurando temas GTK e Plasma para novos usuários

8.3.4 Limpeza Final

Remove a pasta com o código-fonte do tema do diretório '/tmp'. Isso é uma boa prática para não deixar arquivos de instalação desnecessários na imagem final da distro.

```
sudo rm -rf /tmp/Orchis-theme
```

Listing 14: Removendo arquivos temporários

9 Galeria de Imagens da iubaOS

Nesta seção, são apresentadas capturas de tela da distribuição iubaOS em funcionamento, demonstrando o resultado final das personalizações de tema, ícones, papel de parede e a integração dos aplicativos personalizados.

9.1 Tela Inicial e Aparência do Sistema

9.1.1 Tela de Inicialização Boot

A primeira imagem exibe a tela de login personalizada do iubaOS, mostrando o visual de entrada do sistema.

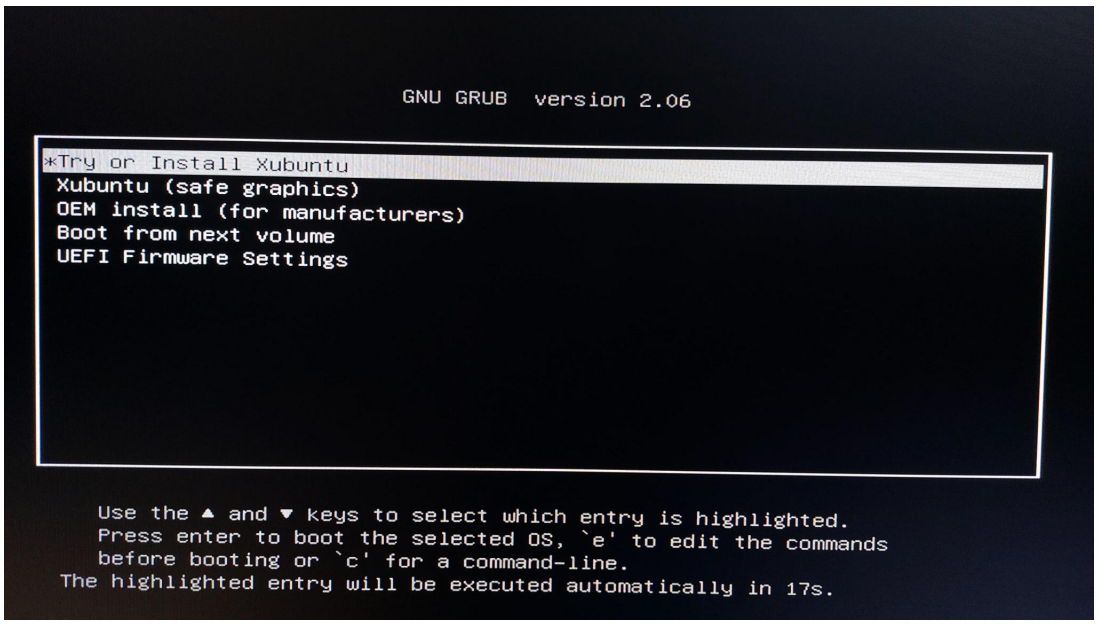

Figura 2: Tela de Inicialização Boot.

9.1.2 Tela de Inicialização imagem .iso

A imagem a seguir exibe uma outra perspectiva ou estágio da tela de login do sistema.

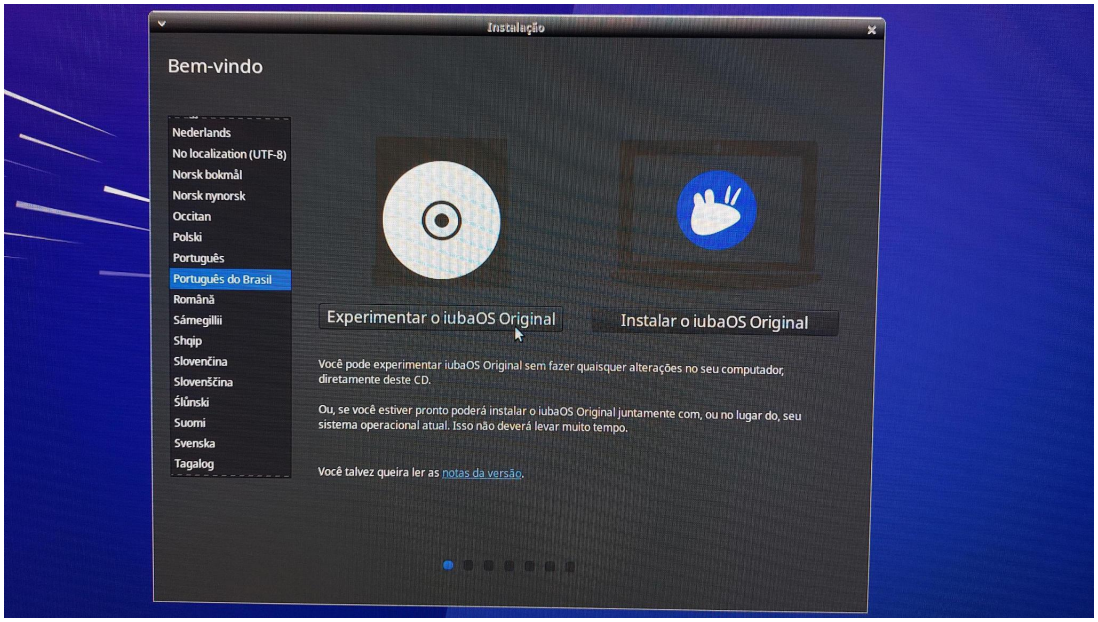

Figura 3: Tela de Inicialização imagem .iso.

9.1.3 Desktop Padrão

Esta captura de tela mostra a área de trabalho padrão do iubaOS após o login, com os elementos visuais iniciais.

Figura 4: Área de trabalho padrão do iubaOS após o login.

9.1.4 Visualização do Tema

A imagem a seguir destaca a aplicação do tema Orchis-Grey-Dark, mostrando as janelas com suas decorações e o estilo visual.

Figura 5: Demonstração do tema escuro Orchis-Grey-Dark aplicado nas janelas do sistema iubaOS, e os icones dos apps.py.

9.1.5 Terminal em Funcionamento

Esta captura de tela exibe uma janela de terminal aberta, demonstrando a interface de linha de comando do iubaOS em uso.

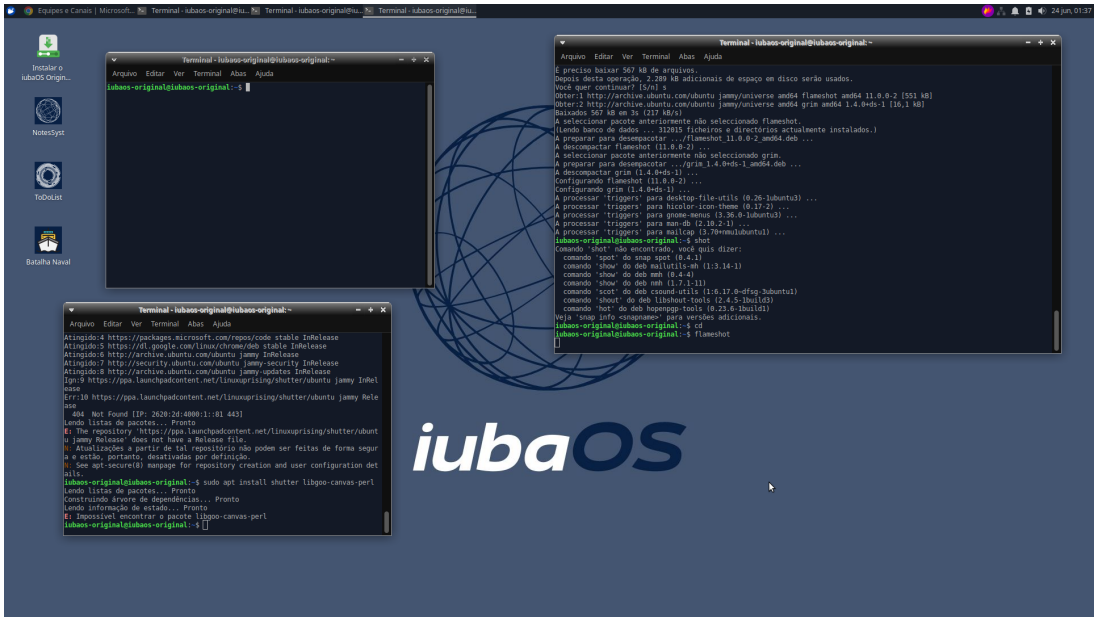

Figura 6: Janela de terminal aberta e em funcionamento no iubaOS.

9.1.6 Papel de Parede Personalizado

A imagem abaixo foca no papel de parede personalizado do iubaOS, elemento chave da identidade visual do sistema.

Figura 7: Detalhe do papel de parede personalizado do iubaOS.

9.2 Aplicativos Personalizados

9.2.1 NoteSyst

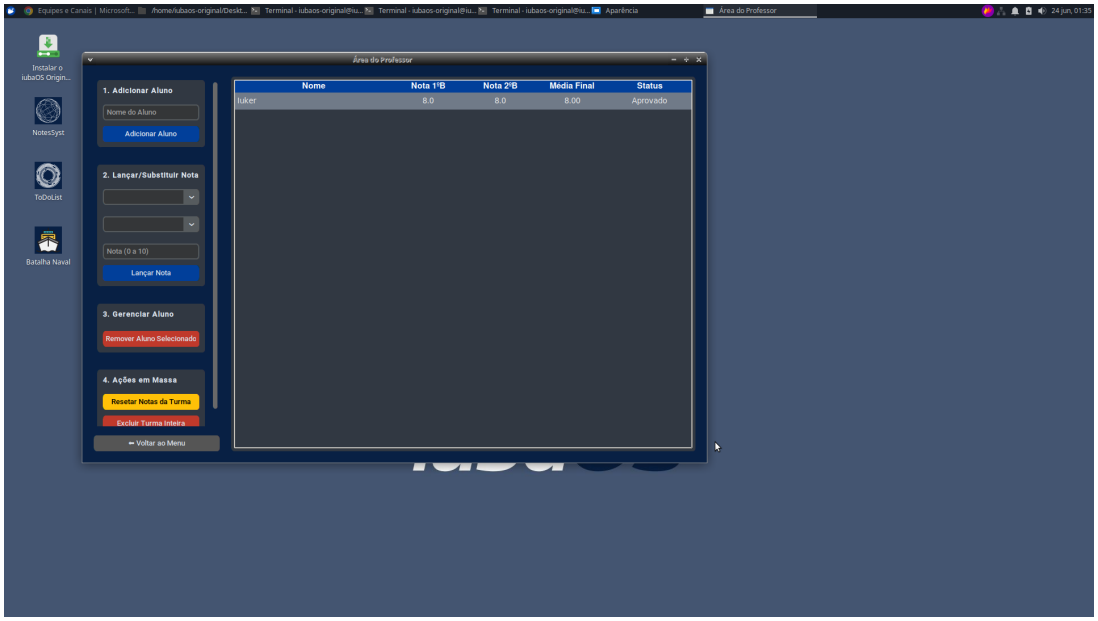

Figura 8: Interface principal do aplicativo NoteSyst, Area Professor.

Figura 9: Interface principal do aplicativo NoteSyst, Area Aluno.

9.2.2 ToDoList

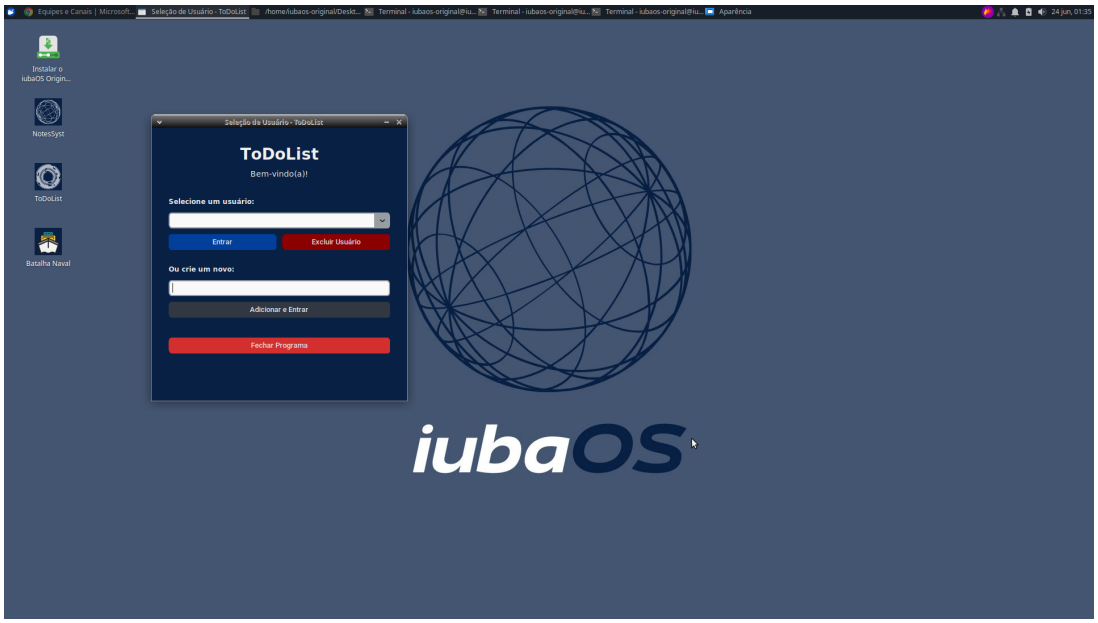

Figura 10: Visão geral da interface principal do aplicativo ToDoList.

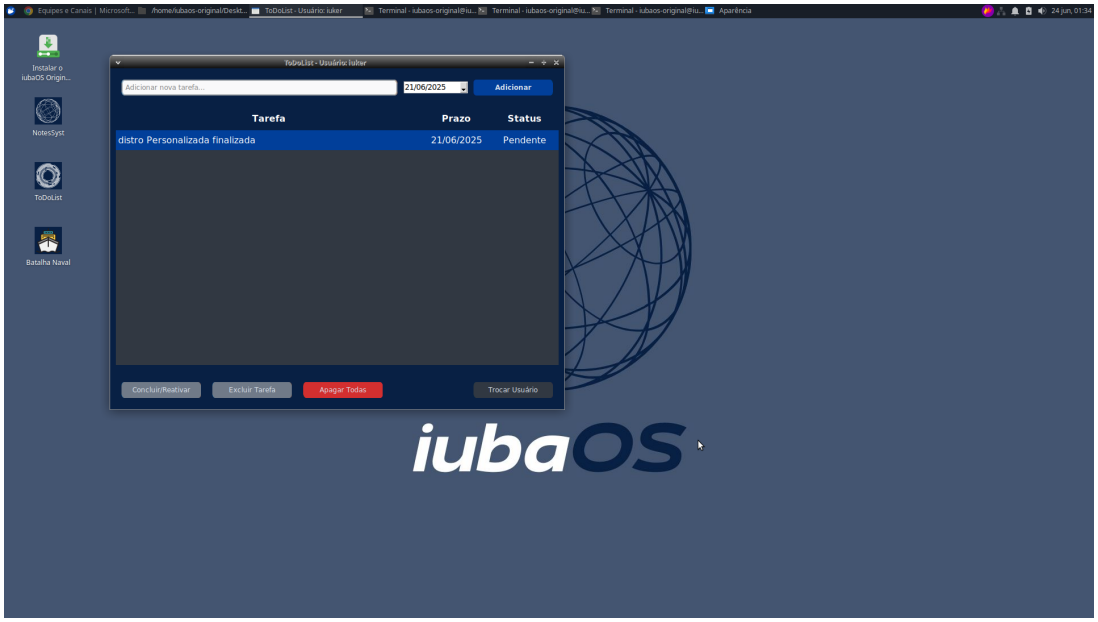

Figura 11: Janela para adicionar uma nova tarefa no aplicativo ToDoList.

9.2.3 Batalha Naval

Figura 12: Tela inicial do jogo Batalha Naval.

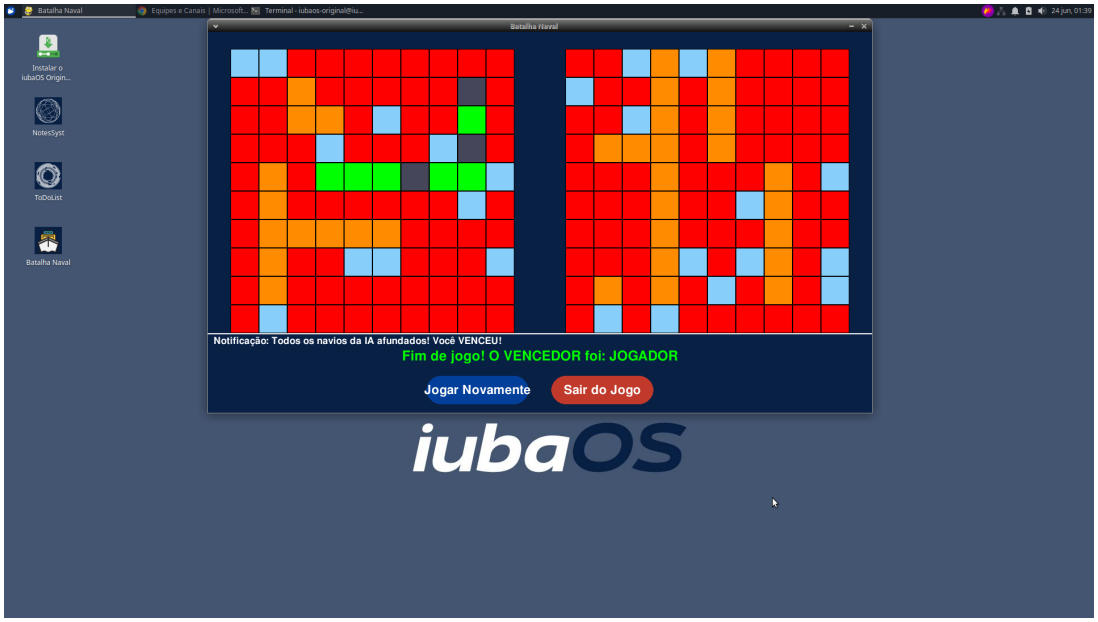

Figura 13: Captura de tela de uma partida em andamento do jogo Batalha Naval.

Agradecimentos e Conclusão Final

Chegamos ao final deste projeto com a satisfação de ter transformado uma proposta conceitual em um sistema operacional funcional. A jornada de criação do iubaOS foi um exercício prático e profundo sobre a arquitetura e a customização de sistemas Linux, superando em muito as expectativas iniciais de aprendizado.

O iubaOS, em sua versão final baseada no Xubuntu, não é apenas um sistema operacional funcional com aplicativos e temas personalizados, mas também um testamento do processo iterativo de desenvolvimento de software: planejar, executar, encontrar um problema, pesquisar, corrigir e repetir. Cada desafio superado representou um conceito de sistemas operacionais aprendido na prática.

Um agradecimento especial ao Professor Baroni pela condução da matéria de Sistemas Operacionais e pela oportunidade de aplicar a teoria na prática ao desenvolver esta Distribuição Linux.

— Fim do Projeto iubaOS —