**UNIVERSIDADE PAULISTA – UNIP EaD**

**Projeto Integrado Multidisciplinar**

**Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas**

**Gustavo Ferreira de Oliveira Coelho – 2465749**

**Projeto de Geração app Empresa**

Diadema SP

2024

**Gustavo Ferreira de Oliveira Coelho – 2465749**

**Projeto de Geração app Empresa**

Projeto Integrado Multidisciplinar em

Análise e Desenvolvimento de Projetos

Projeto Integrado Multidisciplinar para obtenção do

título de tecnólogo em (nome do curso), apresentado

à Universidade Paulista – UNIP EaD.

Orientador (a): Profa. Me. Karhyne Assis

Diadema SP

2024

**RESUMO**

Devido necessidade da startup, a criação de um projeto para auxilia-los devido ao aumento da demanda de trabalhos será elaborado de uma forma coesa e prática de como desenvolver um sistema em linguagem de programação C desde a preparação do ambiente, até o início do projeto, utilização de ferramentas para efetuar testes e desenvolver layouts previamente antes de aplicar ao projeto, download dos arquivos necessários como extensões, frameworks, APIS e um desenvolvedor de interfaces Gráficas para o C, além da utilização de uma IDE para tornar o momento de criação dos algoritmos mais simples e que consiga compilar todas as necessidades da aplicação sem muitos problemas, e no fim criando um ambiente favorável para a empresa destino suprindo a necessidade da mesma, e devido a versatilidade das ferramentas utilizadas e o conhecimento adquirido propiciará ao leitor uma maneira geral de fazer a criação de diversos projetos e suprir outras demandas com estas ferramentas que podem ser utilizadas em múltiplas plataformas.

**Palavras-Chave**: Desenvolver. Projeto. C. aplicação. IDE. Ferramentas. Versatilidade.

**ABSTRACT**

Due to the startup's need to create a project to help them due to the increased demand for work, it will be elaborated in a cohesive and practical way how to develop a system in C programming language from the preparation of the environment, to the beginning of the project, use of tools to perform tests and develop layouts previously before applying to the project, download of the necessary files such as extensions, frameworks, APIS and a developer of Graphical interfaces for C, in addition to the use of an IDE to make the moment of creating the algorithms simpler and that can compile all the needs of the application without too many problems, and in the end creating a favorable environment for the destination company meeting the need of it, And due to the versatility of the tools used, the knowledge acquired will provide the reader with a general way to create various projects and meet other demands with these tools that can be used on multiple platforms.

**Keywords**: Develop. Project. C. Application. IDE. Tools. Versatility.

**SUMÁRIO**

1. INTRODUÇÃO AO PROJETO ................................................................7
2. CAPTURANDO AS NECESSIDADES DO CLIENTE...............................8
3. INTERFACES DE CRIAÇÃO DE LAYOUTS............................................8
   1. FIGMA................................................................................................9-10

3.1.1 COMO USAR?..........................................................................9-10

* 1. GLADE...............................................................................................10-19
     1. INSTALANDO DEPENDENCIAS GLADE E GTK ...................10-11
     2. UTILIZANDO GLADE ..............................................................10-17
     3. CRIANDO E ADICIONANDO CSS..........................................17-19

1. CODE::BLOCKS......................................................................................19-22
   1. BAIXANDO ARQUIVOS NECESSÁRIOS.........................................19-20
   2. DOWLOAD CODE::BLOCKS............................................................20
   3. CONFIGURANDO CODE::BLOCKS.................................................20-22

4.3.1 COMPILER SETTINGS............................................................21

4.3.2 LINKER SETTINGS..................................................................21

4.3.3 SEARCH DIRECTORIES.........................................................21

4.3.4 TOOLCHAIN EXECUTABLES..................................................21

4.3.5 INSERINDO DEBUGGER........................................................22

* 1. CRIANDO PROJETO........................................................................22
  2. CRIANDO ARQUIVO.C.....................................................................22

1. CRIANDO O CODIGO.............................................................................22-41
   1. CRIANDO EVENTOS DE CONTROLE.............................................25-28

5.1.1 CRIANDO EVENTOS DE BOTÃO...........................................25

* + 1. PEGANDO VALORES DE GTKLABEL E INSERINDO VALORES.........................................................25-26

5.1.3 PEGANDO VALORES DE GTKENTRY, GTKENTRYSEARCH E UTILIZANDO SINAL CHANGE..................................................26-27

5.1.4PEGANDO VALOR DE CHECKBUTTON.................................27

5.1.5 PEGANDO VALORES DE SWITCHS......................................27-28

5.2 EXIBINDO JANELAS DE MENSAGEM, ALTERANDO STACK ATUAL E UTILIZANDO GTKFILECHOOSERDIALOG.............................28-30

5.2.1 ALTERANDO STACK ATUAL.................................................28

5.2.2 EXIBINDO E ESCONDENDO JANELA DE MENSAGENS....28-29

5.2.3 UTILIZANDO GTKFILECHOOSERDIALOG..........................29-30

5.3 INSERINDO DADOS EM TABELA (GTK TREE VIEW) UTILIZANDO SINAL ROW ACTIVATED..........................................................................30-32

5.4 RECUPERANDO VALORES DIGITADOS EM GTKTEXTVIEW.........32-33

5.5 UTILIZANDO MYSQL CONNECTOR.................................................33-35

5.6 EFETUANDO REQUISSIÇÕES HTTP COM CURL...........................35-38

5.6.1 EFETUANDO REQUISIÇÃO GET.........................................38

5.6.2 EFETUANDO REQUISIÇÃO POST.......................................38

5.7 EFETUANDO CRIPTOGRAFIA COM LIBSODIUM............................38-39

5.8 CAPTURANDO TEMPO PARA ARMAZENAMENTO DE DADOS.....39-40

5.9 INSERINDO E UTILIZANDO CJSON..................................................40-41

1. CRIANDO CÓDIGO TEORIA DE CÓDIGO...............................................41
2. CONCLUSÃO............................................................................................42

1.**INTRODUÇÃO AO PROJETO**

Uma empresa de soluções ambientais que tem com seu intuito ajudar industrias que precisam abater seus matérias que causam mal ao meio ambiente, precisam devido ao crescimento de sua área de atuação a criação de uma aplicação com a finalidade de facilitar a efetivação do trabalho dito posteriormente, então este projeto será desenvolvido no decorrer deste trabalho e será explicado o passo a passo para a assim atingir a necessidade da empresa.

Nesse processo serão utilizados includes como curl/curl para efetuar requisições para um servidor onde será armazenado as documentações como resíduos tratados e seus respectivos valores, mysql.h utilizada para efetuar requisições a bancos de dados assim podendo fazer uso de dados armazenados em banco, armazenar dados e modificá-los, sodium biblioteca utilizada para fazer a criptografia de dados assim trazendo mais segurança ao usuário final, GTK que é utilizado para criação de interfaces em C que traz uma maior facilidade ao usuário com uma interface totalmente amigável, glib que fazemos uso junto ao gtk para usar certas funções que iram ajudar na manipulação dos dados, time.h que será utilizada para fazer o armazenamento de dados no banco com a data e hora atual do usuário afim de proporcionar uma maior segurança sendo armazenado a data e hora das ações, cJSON que será utilizada para a manipulações de retornos de servidores que normalmente retornam dados em JSON assim podendo utilizar os dados retornados pelos servidores e manipula-los no código, além de includes como string.h para a manipulação de array de caracteres assim conseguindo fazer diversos tipos de manipulação de dados, stdio para manipulação usando FILE, stdlib para alocação dinâmica para memória, ctype.h para funções de verificações de compatibilidade de dados inseridos, mesclando todos esses componentes teremos uma aplicação final que conseguirá satisfazer a necessidade dos clientes e para conseguir utilizar essas mais diversas bibliotecas teremos que utilizar uma IDE que tenha maior compatibilidade com C e suas bibliotecas e o mesmo será o CODE::BLOCKS além do figma que será necessário para a criação de layouts previamente para ter uma maior noção antes de efetuar a criação no aplicativo glade que vem integrado a biblioteca GTK.

Espero que gostem deixarei detalhadamente como cada parte deve ser integrada e ao final no arquivo deixarei um installer que o mesmo contém o projeto já criado todas os arquivos necessários para a compilação além de comandos SQL para inserção dos dados necessários para o banco.

2. CAPTURANDO AS NECESSIDADES DO CLIENTE

Para ser dado o início ao projeto precisamos estar atentos sempre as necessidades pautadas pelo cliente para assim darmos iniciarmos o planejamento, primeiro foi requisitado um sistema onde haja CADASTRO DE USUÁRIOS e CRIAÇÃO DE RELATÓRIOS e sub funcionalidades que tem a necessidade das anteriores para serem criadas corretamente.

Ao ser analisados os processos principais vemos que primeiro temos a necessidade de cadastro de empresas e para isso precisamos de um usuário administrador que poderá realizar todo esse processo e então criamos primeiramente uma tela de login onde caso user ser == admin ele então terá acesso a criação de Relatórios e de usuários e caso user == empresa ele então terá o acesso para os relatórios referente a sua empresa além de mais algumas comodidades que serão criadas posteriormente temos em vista agora que será necessário a criação de uma tela de criação de usuário e uma tela de login com isso em mente daremos o pontapé inicial do projeto que é a elaboração do layout dos mesmos e após isso conseguiremos dar prosseguimento a segunda parte que será uma tela para os administradores gerarem relatórios de empresas e em caso uma empresa acesse a aplicação ela deve ter o acesso estrito somente ao relatório referente a ela podendo realizar o download do mesmo em diversos formatos além de também poder ver um resumo de seus relatórios assim tendo uma noção de quanto foi gasto para o tratamento de resíduo, o quanto de resíduo foi tratado e assim sucessivamente podendo no fim converte-lo e manipula-lo o modelo de conversão utilizado por mim foi o CSV muito utilizado em planilhas de Exel assim os dados podem ser facilmente manipulados pelas empresas para geração de tabelas e resumos de relatórios em geral.

Sabendo dessas informações então daremos o início das criações de layouts necessários e será apresentado a seguir o uso das ferramentas FIGMA e GLADE.

3.INTERFACES DE CRIAÇÃO DE LAYOUTS

A criação de uma interface amigável para o usuário sempre será uma necessidade, pois no seu projeto final o usuário deve conseguir de forma simples e dinâmica, para a necessidade que sua aplicação veio suprir, a seguir serão listados o FIGMA muito utilizado em desenvolvimento web para elaboração de layouts, e por ele ser uma interface de planejamento faz com que no momento de fazer a criação da interface que será inserida no código você já tenha uma noção do que será necessário assim evitando refazer muitas vezes diretamente no desenvolvimento na aplicação em si, e o mais importante o GLADE uma interface desenvolvida para a criação do layout que será inserido junto do código final.

3.1 FIGMA

O figma é uma ferramenta de design onde é amplamente utilizada para criação de protótipos para um produto final onde nele você consegue uma base para seguir no momento de fazer a criação da interface no seu código evitando que aconteça na aplicação a troca no layout por não estar definido fazendo com que seja necessário refazer o mesmo.

3.1.1 COMO USAR O FIGMA?

Por ser uma plataforma online o figma é acessado pelo link: <https://www.figma.com> nele pode ser criado uma conta de forma gratuita a após a criação você será direcionado a uma tela onde a várias informações, para criar um novo design basta clicar em new design file e então você será direcionado para tela de criação de layouts nele há diversas funcionalidades a esquerda estará as suas páginas e layers criados, se necessário a criação de uma nova página basta clicar no + a frente de pages e um pouco mais acima há a aba assets próximo de file, nele você pode adicionar bibliotecas com ícones e alguns layouts prontos assim facilitando a criação, já na aba abaixo haverá diversos ícones o primeiro é um ícone de seleção o segundo é para frames onde você pode inserir alguns elementos no design já o terceiro é para criação de formas nele você pode inserir formas geométricas, linhas e imagens já o quarto é para criação de desenhos com o mouse para utiliza-lo é recomendado uma mesa digitalizadora para facilitar a criação, já o quinto é para a criação de caixas de texto, o sexto é para casos de mais pessoas simultâneas estiverem trabalhando no projeto onde permite adicionar comentários em partes criadas, assim facilitando para um outro designer, já o sétimo é para a inserção de componentes como o assets comentado mais acima já a última aba seria para usuários premium nesta parte não será necessário e no canto direito haverá a aba de personalização dos componentes onde é possível corrigir coloração dos componentes e suas respectivas configurações, bom sabendo desses usos agora é possível criar seu layout no figma e ele será a base para a criação na interface glade.

3.2 GLADE

Primeiro o que necessitamos saber do GLADE, ele é um criador de interfaces visuais que vem em conjunto para tornar a programação em GTK mais simples, onde o Dev não necessita desenvolver a interface por meio de códigos mais complexos, porém Glade foi criado especificamente para essa finalidade não sendo muito maleável quando se trata de plataformas sem o uso do GTK o mesmo gera um XML que contém os layouts criados pelo usuário e utilizando a propriedade gtk\_builder\_new assim carregando o xml criado.

3.2.1 INSTALANDO DEPENDENCIAS GLADE E GTK

Bom vamos ao primeiro passo para efetuar a instalação do glade e em conjunto a biblioteca GTK para isso será necessário efetuar a instalação do primeiro componente no Windows o msys2 ele é um bash onde será possível fazer as instalações necessárias, nas referências estará disponível o site onde eu tomei como ponto de partida no site: <https://www.msys2.org> você encontrará o instalador necessário ao efetuar o download abra o bash MSYS2 MSYS e nele escreva a sequencia de comandos para pacman -Syu ele fará com que sejam sincronizados os pacotes atualizados e então aparecerá uma pergunta para prosseguir com a instalação, basta escrever “Y” e apertar ENTER e então a instalação será efetuada, após isso utilize o bash novamente e utilize o próximo comando que será para instalar ferramentas básicas para o GTK digite o seguinte comando no bash “pacman -S mingw-w64-x86\_64-toolchain” assim serão listados diversos repositórios com as bases necessárias você pode escolher somente os necessários presente no site da referencia ou fazer como eu e instalar todas que seria a opção default, a única questão de efetuar a instalação de todos os componentes é o uso maior da memória do seu sistema, por isso verifique sempre o que será melhor para sua necessidade, após selecionar os necessários basta aguardar ele perguntar se deve prosseguir com o download basta digitar “Y” e apertar ENTER a aguardar a instalação dos pacotes, agora será necessário baixar algumas ferramentas para compilação elas serão necessárias para quando estivermos configurando o CODE::BLOCKS e para compilações via bash se necessário então digite o seguinte comando pacman -S base-devel com as mesmas questões do anterior serão enumerado diversos downloads você pode selecionar todos ou apenas os necessários após selecionar basta escolher se quer prosseguir e efetuar o download dos compiladores, agora com as ferramentas baixadas deve ser feito o comando pacman -S mingw-w64-x86\_64-gtk3 para efetuar o download do GTK o procedimento é como dos anteriores e agora a ferramenta muito importante que nos será necessário o GLADE basta digitar no bash o comando pacman -S mingw-w64-x86\_64-glade e assim o GLADE será instalado após feito esse procedimento verifique na pasta criada normalmente na C: o nome é msys64 nela haverá diversas pastas usaremos a pasta home para criar o ambiente de código então dentro dele crie a pasta para o projeto e então volte para a pasta msys64 e entre na pasta mingw64 nela há os componentes do GTK além de também o GLADE, ele fica inserido dentro da pasta bin com o nome de glade.exe, para facilitar crie um atalho na sua área de trabalho com o mesmo, assim tornando mais fácil executa-lo sem a necessidade de retornar a pasta bin a todo momento.

Página: 8

3.2.2 UTILIZANDO GLADE

Após todos os procedimentos anteriores podemos então iniciar um projeto GLADE o mesmo deve ser com base no layout criado anteriormente no FIGMA ou um simples para testes, irei explicar diversos pontos para que você ao desenvolver tenha um bom ponto de partida sabendo sobre os componentes, criação, testes e até mesmo a integração de um CSS personalizado.

Ao abrir o app você terá uma tela com a parte somente a parte de cima da janela para efetuar certas ações, as ações descritas a seguir estão sendo referenciadas do canto superior esquerdo até o canto superior direito, como abrir um projeto presente em sua máquina, criar um novo projeto, desfazer e refazer, salvar o projeto, propriedades do projeto, nesta opção onde podemos inserir um CSS, e uma Dropbox onde há opções de configuração de salvamento, introdução entre outros.

A primeira ação será clicar em novo projeto para assim darmos início a criação do layout, o primeiro sempre será a criação da janela, para isso deve se clicar NIVEIS SUPERIORES e então se abrirá um Dropbox.

Janelas:

As janelas utilizadas por mim para a aplicação foram a GTKApplicationWindow que cria a janela principal para seu app, a que será exibida primeiramente ao Usuáro, GTKMessageDialog essa será uma janela de mensagem que sobrepõem a janela principal, assim podendo se criar mensagens de confirmação, mensagens de erro e também para criar uma janela simples onde o usuário pode executar certas ações e para algo mais especifico como selecionar o local de Download dos relatórios podemos utilizar a janela GtkFileChooserDialog uma janela que mostra os arquivos da máquina do usuário podendo então ser criado um local onde é possível selecionar o local para Download, todos esses itens ao serem selecionados no canto direito tem suas opções de layout como definir o ícone que ficará visível ao abrir a janela o nome da janela a altura e largura padrão da janela entre muitos outros citarei aqui as que são principais, primeiro o ID essa barra onde se digita o ID é o nome que devemos chamar no código C para manipularmos esse componente via Código, na aba superior selecione o Comum nele tem um input escrito NOME DO COMPONENTE nele é digitado o nome do componente para conseguirmos manipula-lo via CSS, TODO NOME DE COMPONENTE EM UM CSS É VISTO COMO UM ID DE HTML, e na aba SINAIS após comum temos diversas opções e cada uma delas é uma ação que pode ser inserida no componente para utilizarmos esse sinal no código fazendo com que podemos realizar alguma ação referente a esse sinal, citarei os sinais importantes de cada componentes por partes, como estamos a princípio estamos falando das janelas citarei alguns muito necessários em suas respectivas janelas.

Na janela principal o seu GTKapplicaionWindow é importante ir na aba GtkWidget e inserir um evento destroy, nesse evento quando utilizado ele faz a destruição da janela ou seja ao usuário clicar, por exemplo, no X para fechar a janela a mesma seja destruída sem a intenção de ser aberta novamente, utilizamos mais na janela principal e caso ela seja fechada todo a aplicação irá parar, para inserir esse evento clique duas vezes na aba DIGITE AQUI na COLUNA MANIPULADOR, elas estão visíveis no topo abaixo das abas comum, sinais, empacotamento e geral, ao clicar duas vezes digite ON\_ID na parte ID digite o id criado para a aplicação ao ser feita a digitação do ON o mesmo já aparecerá o nome do ID definido então basta clicar duas vezes e ele auto completará e ao ser digitado aperte ENTER e então uma nova linha será inserida abaixa indicando que a ação foi inserida corretamente.

Já na GtkMessageDiolog não é necessário um sinal especifico, pois ela deve ser escondida e não destruída, pois ao ser destruída ela não abrirá novamente a não ser que seja chamado novamente do zero, porém o nome da janela e ícone pode ser definido o ID ainda sim será necessário e o CSS caso você ache necessário.

Já na GtkFileChoicerDialog é necessário na parte de sinais definir o sinal de response em GtkDialog isso fará com que poderemos dependendo do que o usuário escolher como selecionar uma pasta retornar o caminho selecionado além de caso aja necessário criar um elemento css e não esquecer do ID.

Após a configuração das janelas devemos agora fazer a inserção de elementos para moldarmos o layout da página, explicarei sobre a aba contêiners.

Contêiners:

Os contêiners são elementos muito importantes para a criação, ele é o que insere os espaçamentos na página para inserirmos componentes, como botões, textos, inputs entre outros dentre os diversos contêiners irei destacar 3 que são de suma importância.

O primeiro contêiner é o GtkGrid esse elemento faz uma divisão em formato de jogo da velha onde o dev pode em cada um desses espaços definir uma linha ou uma coluna e na aba empacotamento você pode ou diminuir ou aumentar o número de divisões e em cada espaçamento pode ser inserido um novo Grid assim você pode aumentar os componentes em um espaço mais limitado de outro Grid no mesmo pode ser inserido um Nome para manipulação em CSS entre outras configurações.

O segundo contêiner é o GtkScrolledWindow ele faz com que quando o elemento dentre dele chegue ao limite da tela ou limite de altura e largura definido ele insere uma barra de rolagem para assim não sobressair o espaçamento de tela.

E o terceiro contêiner é o GtkStack ele faz com que seja possível criar diversas visualizações em um espaçamento já inserido, ele é normalmente utilizado para criar como se fossem abas de ao usuário clicar em algo ele insere uma nova tela que sobressai a antiga, assim não precisando criar diversas novas janelas, para definir o tanto de janelas que irão ser criadas e qual está sendo editada fica na aba GERAL em Editar Página e Número de Páginas nele é necessário a inserção de um ID para manipulação das janelas por meio de eventos e pode ser adicionado uma transição de um layout para o outro “AVISO IMPORTANTE” ao ser criados telas dentro do Stack cada tela haverá a opção NOME na aba EMPACOTAMENTO esse nome é necessário para fazermos a troca entre janelas dentro do código, e ao ser criados abas no Stack a aba visualizável sempre será a página 0 por isso ao criar itens em abas diferentes você deve sempre ir em empacotamento na tela que está sendo criada dentro do Stack e defini-la como página 0 assim podendo visualizar pelo elemento citado abaixo”.

Todos os elementos ao serem criados podem ser visualizados no canto superior esquerdo ao lado de uma lupa o símbolo de engrenagem, nele abrira o item em questão assim podendo visualizar com detalhes o resultado atual da janela.

Agora falaremos sobre a aba Controle.

Controle:

Na aba Controle ficam os componentes que capturam eventos, como um botão, um link, um botão toggle onde o usuário pode ativar e desativar, inputs para digitação de texto do usuário entre outros, os que estarei explicando serão os botões, as entradas de texto, CheckButton, Switch e LinkButton.

Começarei falando do GtkButton primeiro e mais importante para todos os componentes que devem ser manipulados devem ser inseridos um ID de inicio e no caso do Button ao ir a aba sinais haverá a dropox que contem GtkButton como principal, cada um dos componentes de controle tem seus sinais únicos para o controle em especifico o mais usado para o botão é um clicked, pois é um evento que captura os cliques do usuário e executa uma função no qual ele está vinculado, para inserir o evento basta clicar na aba manipulador digitar on e ele auto completará com o ID e então digitar ENTER e então uma aba vazia irá aparecer isso indica que funcionou e o evento está cadastrado já para decoração a aba geral e comum fazem as manipulações teste o que mais lhe convém, o css será explicado em um capítulo separado.

Agora o GtkEntry e GtkEntrySearch ambos são semelhantes, porém o GtkEntry é mais recomendados para entradas de usuário como cadastro e assuntos que serão salvos para uso posterior já o GtkEntrySearch é mais recomendado para pesquisas em si, para a busca de informações e dados dentro da aplicação referente as entradas na aba sinais após definir um ID um sinal interessante para uso seria o Changed, pois nele a cada digitação de usuário executa uma ação ele fica em GtkEditable, a diferença no sinais do GtkEntrySearch é que a opção Changed está com o nome de Search-changed.

Página: 11

Agora o GtkCheckButton nele é criado uma caixa onde pode ligar e desligar sua ação e o mesmo retorna um valor de boolean true e false para inserir o sinal necessário basta ir em GtkToggledButton e inserir o sinal toggled.

Agora em um GtkSwitch é interessante ser utilizado um dos dois sinais inseridos o activate e o state-set o activate é ativado em caso o switch foi ativado já o state-set é acionado cada vez que que o sinal é alterado como na CheckButton retornado true em caso acionado e false em caso desativado.

Já o LinkButton são textos que ao serem clicados executam ações normalmente utilizados para inserir links de páginas web para inserir um link é só digitar o URL abaixo de Id e ao usuário clicar ele será direcionado a pagina inserida.

Exibição:

Vamos então falar sobre a aba Exibição, onde podem ser inseridos elementos mais voltado para decoração, porém nessa aba há elementos muito importantes, principalmente para contato com o banco de dados, primeiro o GtkLabel são as partes textuais que são inseridas em seu layout, GtkAccelLabel são semelhantes ao GtkTextLabel, GtkImage faz com que seja possível inserir imagens no app, GtkTextView nele pode ser inseridos textos do usuário contendo quebra de linhas e suporte maior a caracteres, GtkTreeView um elemento muito importante e maleável são as tabelas nelas podem ser inseridos um grupo de informações de forma simples e pratica, os demais serão menos utilizados, porem algo que incrementa bastante no layout de downloads são as barras de progresso que contem nesta aba, agora serão explicados com mais detalhes os elementos acima.

O GtkLabel e GtkTextLabel são utilizados para o dev efetuar a decoração das páginas inserir explicações previamente carregadas pelo usuário, mas também manipulando via Id é possível fazer com que esses labels insiram informações Dinâmicas ao usuário, para isso basta inserir um ID e eles poderão ser manipula-los.

GtkImage é feito pra fazer a inserção de imagens podendo ser inserida junto a botões, em layouts como no header de páginas entre outros.

GtkTextView utilizado para criação de textos do usuário, diferente do label onde podendo limitar as informações digitadas e não aceitando quebras de linha, o GtkTextView é feito para suprir essas necessidades do GtkEntry e esse texto pode ser enviado para um banco ou um local que esteja utilizado para o armazenamento.

GtkTreeView o GtkTreeView é utilizado para manipulação de dados em formato de tabela, assim sendo muito maleável para inserção de dados um sinal interessante a ser inserido na mesma é o row-activated que quando o usuário clica duas vezes na linha selecionada e coluna, o mesmo pode executar uma função especifica para utilizar o GtkTreeView é necessário efetuarmos algumas etapas para seu funcionamento correto, primeiro após inserirmos um GtkTreeView na aba Geral há escrito abaixo MODELO DE TREEVIEW clique no lápis ao final e abrirá uma aba objetos, esses objetos são liststores neles que ocorrem a inserção de dados e exibidas do TreeView você pode selecionar um existente caso queira duas tabelas com as mesmas informações ou criar um novo, para criar basta clicar em Novo na aba de botões, e então ao olhar na parte esquerda onde fica armazenado as janelas que estão sendo trabalhadas haverá um liststore1 clique no mesmo e na aba a esquerda em geral você poderá inserir os valores das colunas e seus respectivos nomes para criar uma nova coluna clique duas vezes em DEFINA UMA NOVA COLUNA e então aparecerá diversas opções há duas que são consideradas principais em minha opinião que é o gchararray ele tem suporte para uma cadeia de caracteres assim podendo armazenar dados escritos com uma grande quantidade de letras e o GtkPixBuff nele ao ser criado essa coluna pode ser encaminhando um ícone presente no caminho escrito, após definir o tipo de coluna deve-se decidir o nome da coluna, após finalizar o Liststore volte no GtkTreeview clique com o botão direito e selecione a opção editar ao clicar em editar abrirá uma nova janela, nela você pode clicar nos botões + para adicionar uma coluna e – para retirar uma coluna adicione uma coluna e a direita aparecerá as opções de layout da coluna o interessante de ser alterado é o Titulo ele recebe o nome da coluna, ao se clicar com o botão direito na coluna criada você pode definir o valor que será inserido nessa coluna os que utilizei é ADICIONAR FILHO TEXTO e ADICIONAR FILHO PIXBUF ao adicionar o filho texto aparecerá a direita diversas opções, essas são as formatações do texto -1 é o valor padrão, você pode definir manualmente as formatações desativando o CheckButton e com o mesmo ativado clique na Dropbox não definido e nele aparecerá as colunas definidas no liststore para estilo de texto defina a coluna que contenha o valor gchararrey e em caso de ser um filho pixbuf adicione a coluna que recebe valor de pixbuf e assim pode ser adicionado valores no liststore e serão exibidos na tabela.

Algumas observações é que ao se clicar com um botão direito em qualquer um dos componentes que ocupam um grande espaço selecione a opção adicionar pai e você pode definir tipos de valores acima desse componente e em caso ser inserido janela de rolagem por exemplo em uma GtkTreeView o mesmo ao avançar para o espaço acima do limite inserido na janela de rolagem ele irá criar uma rolagem automática assim trazendo uma beleza maior ao projeto.

3.2.3 CRIANDO CSS

CSS é um arquivo de estilização normalmente utilizado para páginas HTML, porém o Glade tem suporte a certos comandos CSS que podem estilizar de forma interessante os componentes de seu layout.

A melhor IDE para fazer a estilização de forma fácil e completa eu recomendo o download do Visual Studio Code, pois ele além de uma interface simples para uso, ele já a um debug para códigos em css que auxiliam na escrita e em tratativas de erro para efetuar o download basta acessar a página <https://code.visualstudio.com> e então baixar o executável para assim instalar em sua máquina.

Para criar um arquivo css é necessário que o arquivo seja digitado com a extenção .css no final do nome do arquivo, ou seja, ao criar um arquivo faça a seguinte esquematização nomedoarquivo.css e assim será criado o arquivo CSS, em css para fazer a estilizações, você deve inserir o Nome do Componente com # no início e { abrir e fechar chaves } e dentro das chaves inserir os comandos que vão fazer com que manipule seus componentes como, fundo, texto e espaçamento, todo comando css deve ser finalizado com ; após cada comando, agora explicarei certas estilizações que são compatíveis com o Glade e utilizando você poderá deixar sua aplicação mais bonita e bem formada.

Formatações de Texto:

Para formatar textos em css, pode se usar os seguintes comandos:

font-size: tamanhodafontepx; este comando ao ser substituído o texto tamanhodafonte por um número decimal ele fará com que a fonte atual do componente receba o seu tamanho em pixels.

font-weight: expessuradafontepx; este comando fará com que ao substituir o texto expessuradafonte por um número decimal ou o texto bold aumentará a expessura da fonte, ou seja, ela terá um contorno maior.

color: nomedacor; este comando fará com que ao ser inserido a um texto ele alternará a cor pela desejada, obs as cores tem que serem digitadas em rgba ou o nome de uma cor em inglês.

Formatação de Backgrounds:

Agora para formatar os backgrounds da sua aplicação via css, pode se usar os seguintes comandos:

background-color: cordefundo; esse comando faz com que o fundo do elemento fique da cor inserida em cordefundo.

background-image: url(“./local/da/imagem.png”); esse comando faz com que você consiga inserir uma imagem de fundo inserindo a localização exata da imagem na máquina, é recomendado que se crie uma subpasta img junto do css para assim o caminho ser simplificado como “./img/nomedoarquivo.jpg”.

Espaçamentos:

No css é possível definir um espaçamento maior para o componente e também é possível fazer um espaçamento fora do componente que interfere na modelagem do mesmo.

Padding: tamanhodocomponentepx; isso faz com que seja adicionado um espaçamento do componente atual com o componente pai aumentando o atual e adaptando o pai, padding é possível se utilizar de diversas formas ao ser digitado apenas um tamanho todas as bordas terão o mesmo espaçamento, por exemplo, padding:20px; significa que todos os lados terão o espaço de 20 pixels caso seja adicionado mais um valor, por exemplo, padding:20px 15px; este significar que o eixo y, ou seja o em cima e em baixo haverá um espaçamento de 20 pixels já o eixo X, esquerda e direita terão o espaçamento de 15 pixels e por fim caso sejam utilizados padding: 10px 5px 15px 20px; este comando dessa forma definirá um tamanho diferente para cada lado do componente, sendo 10px para o topo, 5px para a direita, 15px para baixo e 20px para a esquerda.

Margin: tamanhodamargempx; essa propriedade faz com que no componente pai seja adicionado um espaçamento no componente atual, assim fazendo com que ele possa ser deslocado entre o espaçamento do componente pai, assim como no padding ele tem as mesma propriedades de espaçamento sendo margin:20px; será adicionado esse espaçamento em todos os lados, margin:20px 15px; adicionando espaçamentos no eixo X de 15px e no eixo y de 20px e ao se utilizar margin:10px 5px 20px 15px; será adicionado um valor diferente em cada lado, referente ao margin é possível se adicionar no próprio GLADE na parte Comum haverá a opção margens e pode se adicionar em cada lado do componente.

4.CODE:BLOCK

Agora que finalizamos a criação do Layout chegou a hora de utilizarmos uma IDE combatível com C que contém a melhor forma de fazer a junção das diversas bibliotecas que utilizaremos, a IDE escolhida para tal será o CODE::BLOCKS.

Uma primeira ação importante é ir na pasta home em msys64 e criar uma pasta onde será inserido os arquivos do projeto como pasta de imgs e arquivos como Css e apk.glade com o seu layout criado.

4.1 BAIXANDO ARQUIVOS NECESSÁRIOS

O primeiro Arquivo necessário será o mysql-connector ele é o responsável pela comunicação do código com o banco de dados no instaler disponibilizado junto do zip do projeto na pasta que será criada adicionará o mysqlconnector, mas caso seja necessário o Download em página ele está diponivel em <https://downloads.mysql.com/archives/c-c/> basta selecionar o arquivo ZIP Windows para seu tipo de PC caso seja 64 bits baixe a seguinte Windows(x86, 64-bit),ZIP Archive e caso 32 é o download abaixo, ao finalizar o Dowload e extraia a pasta insira a pasta descompactada na pasta do projeto.

Agora o Segundo será o cJson ele também estará na pasta do installer, porém ele é possível ser baixado em um repositório de GtiHub no link a seguir <https://github.com/DaveGamble/cJSON> para fazer o Download basta clicar em code e selecionar Download ZIP extraia o na pasta do seu projeto.

O próximo item será o libsodium um componente que utilizaremos para criptografia de dados, para baixa-lo, abra o bash msys64 minGw64 e digite o seguinte comando pacman -S mingw-w64-x86\_64-libsodium e o mesmo será incluído.

O seguinte item será utilizado para as requisições HTTP o libcurl, para instala-lo basta seguir o mesmo procedimento anterior e digitar no bash do mingw64 pacman -S mingw-w64-x86\_64-curl e o mesmo será incluído.

Estás são as bibliotecas que são necessárias a instalação o restante estará automaticamente incluído.

4.2 DOWLOAD CODE::BLOCKS

A versão utilizada por mim foi a versão 17.12 e a mesma pode ser baixada no link <https://sourceforge.net/projects/codeblocks/files/Binaries/17.12/Windows/codeblocks-17.12mingw-setup.exe/download> ela está disponível para Download para Linux e Mac caso queira efetuar o download de novas versões para o Windows para o Linux e Mac basta acessar o link <https://www.codeblocks.org/downloads/binaries/#imagesoswindows48pnglogo-microsoft-windows> e escolher a plataforma e a versão, após o Download basta acionar o executável e realizar o Download em Choice Component é bom que seja selecionado todos, para efetuar diversos tipos de compilações.

4.3 CONFIGURANDO CODE::BLOCKS

Aqui serão explicados como configurar seu CODE::BLOCKS para compilar e conseguir debugar seu código sem problemas.

De início ao abrir a IDE clique em Settings que fica nos cabeçalhos ao ser selecionado ele abrirá algumas opções, de inicio clique em compiler verifique se em SELECTED COMPILER se o compilador selecionado é o GNU GCC Compiler caso não seja clique e selecione o mesmo que deve estar disponível, após ser selecionado verifique as janelas abaixo presentes vamos primeiro ao Compiler Settings.

4.3.1 COMPILER SETTINGS

nele devemos ir a aba OTHER COMPILER OPTIONS nele digite os seguintes valores `pkg-config --cflags gtk+-3.0` agora digite ENTER caso não quebre uma linha e então digite o segundo `pkg-config --cflags --libs gtk+-3.0` para caso erros.

4.3.2 LINKER SETTINGS

Em Link Libraries nesta aba deve ser linkado o dll do mysqlconnector para o uso do comunicador entre código e banco de dados, em link libraries clique em ADD clique no ícone de pasta e então entre na pasta no mysql-connector criada na pasta do projeto em Home entre na pasta LIB e selecione libmysql.lib basta seleciona-lo, então na aba OTHER LINKER OPTIONS digite o seguinte comando `pkg-config --cflags --libs gtk+-3.0` está aba então está completa.

4.3.3 SEARCH DIRECTORIES

Na aba COMPILER clique em ADD e então adicione as pastas include que está presente na pasta do mysql-connector e após ela adicione a pasta include presente em uma subpasta do msys64 ao entrar na pasta msys64 selecione a pasta mingw64 selecione a pasta include agora ainda em SEARCH DIRECTORIES entre na aba LINKER e selecione as pastas lib e bin do mysql-connector e a pasta lib do mingw64 após isso a aba SEARCH DIRECTORIES está pronta.

4.3.4 TOOLCHAIN EXECUTABLES

Na Compilers installation você pode clicar em auto-detect para ver se ele encontra o caminho dos arquivos necessários, mas caso seja necessário clique em ... e selecione o caminho da pasta mingw64 após seleciona-la clique em OK e após em PROGRAM FILES na primeira opção C compiler: deve se adicionar o GCC.EXE para encontra-lo clique nos ... e entre na pasta bin do mingw64 e busque o gcc.exe, em C++ compiler: selecione o g++.exe também na pasta bin, em Linker for dynamic libs: insira nele o g++.exe também, em Linker for static libs selecione ar.exe na pasta bin, em Debugger selecione o GDB/CDB debugger : Default, em Resource Compiler seleceione windres.exe, na pasta bin e em Make Program será necessário ir na pasta user em msys64 entre na pasta usr e selecione a bin e então copie o arquivo make.exe e cole na pasta bin do mingw64 e então insira em Make Program.

4.3.5 INSERINDO DEBUGGER

Após finalizada a etapa anterior clique em OK confirme se as alterações foram salvas e então novamente em SETTINGS clique e entre na aba Debugger e então basta clicar em Default caso ele não esteja aparecendo clique no sinal de + presente em GDB/CDB debugger e ele aparecerá em Executable Path insira o arquivo gdb.exe presente na pasta bin do mingw64 e o restante dos dados podem ser mantidos clique em OK e eu recomendo que após essas configurações o CODE:BLOCKS seja reiniciado e verifique se os valores forma mantidos.

4.4 CRIANDO PROJETO

Para criar um projeto em C na tela inicial do Code Blocks clique em Create a new Project e na tela de template selecione Empty Project no primeiro input insira um tiltulo para o projeto o segundo item escolha a pasta criado para o projeto em Resulting Filename ele costuma criar um diretório com o nome do projeto, apague o diretório em especifico e então ele irá ser instalado no diretório Selecionado anteriormente, após isso basta verificar se o compilador selecionado é o GNU GCC COMPILER, caso não seja selecione-o, o restante das opções podem ser mantidas, clique em finish e o projeto será criado.

Ao criar o projeto antes de dar inicio a uma file em C precisamos incluir alguns itens, na barra esquerda onde fica o nome do seu projeto clique com botão direito no nome e então clique em Build Options e na janela criada vá para LINKER SETTINGS e em link libraries clique em ADD e adicione o arquivo libsodium.a e libcurl.dll.a que serão encontrados dentro da pasta lib em mingw64, após essa configuração clique mais uma vez em seu projeto com o botão direito e seleciona ADD FILES e então vá a pasta do cJSON e selecione o arquivo cJSON.c assim o arquivo para interpretar dados json, então após esses procedimentos estamos prontos para dar início há criação da codificação back end em C.

4.5 CRIANDO ARQUIVO.C

Para criar o arquivo em C dentro do Projeto basta estar com o Projeto aberto e então clicar no primeiro ícone abaixo do cabeçalho ele estará com o ícone de um papel com um + ao lado clique nele e selecione Empty File.

5. CRIANDO O CODIGO

Agora daremos inicio a criação do código em C após todas essas etapas, primeiro devemos fazer todos os includes necessários, primeiro devemos incluir primeiro vamos incluir o GTK para incluirmos usamos #include <gtk/gtk.h> assim o Gtk será incluído ao seu código, incluiremos em seguida o glib use o código #include <glib.h> ele vem incluso com o GTK, porem ele trás algumas funções uteis para manipular dados, após vamos incluir o msql conector use o código #include <mysql.h> agora vamos incluir bibliotecas que já vem por padrão sendo elas stdio.h, stdlib.h e string.h basta fazer o #include <stdio.h>, #include <stdlib.h> e #include <string.h>, atualmente o código estará da seguinte forma.

#include <gtk/gtk.h>

#include <glib.h>

#include <mysql.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <curl/curl.h>

#include <sodium.h>

Para compilarmos uma página teste que deve carregar toda a interface gráfica criada digite o seguinte código

GtkBuilder \*builder;

int main(int argc, char \*argv[])

{GtkWidget \*window;

GtkStack \*stack;

gtk\_init(&argc, &argv);

builder = gtk\_builder\_new();

if (!gtk\_builder\_add\_from\_file(builder, "./glade/apk.glade", NULL))

{ g\_printerr("Erro ao carregar arquivo de interface XML.\n");

return 1;}

window = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "Janela\_Principal"));

gtk\_widget\_show\_all(window);

gtk\_main();

return 0;}

Vou aqui explicar cada etapa do código, primeiro definimos os includes sempre no cabeçalho do código abaixo dos includes definimos variáveis globais, onde todas as funções utilizaram os valores inseridos nessas variáveis, neste teste utilizamos apenas a função main para abrir a janela criada, após os includes foi criada o ponteiro builder, este builder é o construtor em que inserimos o xml criado pelo glade, então nele está armazenado os valores que vamos manipular para inserir suas respectivas funções, agora dentro da janela main criamos os ponteiros window que receberá um Widget e um Stack que recebera o seu gtkstack criado, utilizamos o gtk\_init dentro dele recebe os valores dos parâmetros da função main e então fazemos com que o ponteiro builder receba um gtk\_builder\_new() e dentro do if que retornará TRUE caso o arquivo glade seja encontrado e NULL caso não o ponto de exclamação inverte os valores então caso exista ele retornará FALSE, utilizamos dentro do IF a função gtk\_builder\_add\_from\_file(builder,”caminho/do/arquivo.glade”,retorno em caso de erro) dentro do IF inserimos um g\_print da biblioteca glib, ele funciona como um printf, porem feito especialmente para o console GTK e retornamos 1 para a aplicação ser fechada e então caso a aplicação continue em pleno funcionamento inserimos o valor do id Janela\_Principal em window dessa forma a janela principal criada no glade tem seu id retornado por essa linha e então chamaos um gtk\_widget\_show\_all(window); esse valor mostra para o usuário a janela principal e então chamado o gtk\_main() para mostar que é o valor da janela main e então retornamos 0.

Este código irá compilar a janela da aplicação criada sem o CSS, para carregarmos do mesmo devemos incluir a seguinte função:

void carregar\_css(void)

{GtkCssProvider \*provider = gtk\_css\_provider\_new();

GdkDisplay \*display = gdk\_display\_get\_default();

GdkScreen \*screen = gdk\_display\_get\_default\_screen(display);

gtk\_css\_provider\_load\_from\_path(provider, "./glade/css/Projeto.css", NULL);

gtk\_style\_context\_add\_provider\_for\_screen(screen,GTK\_STYLE\_PROVIDER(provider), GTK\_STYLE\_PROVIDER\_PRIORITY\_USER);

g\_object\_unref(provider);}

está função é a função necessária para carregarmos o css em nossa aplicação, primeiro precisamos declarar um ponteiro provider especificamente para carregamento de css, o mesmo recebe o valor de gtk\_css\_provider\_new(); após isso declaramos um ponteiro display para receber o display\_default(); e então uma variável screen que receberá o tela atual tendo como parâmetro o display e então inserimos em gtk\_css\_provider\_load\_from\_path(provider criado, caminho/do/arquivo.css,valor de retorno em caso de erro); e então em gtk\_styke\_context\_add\_provider\_for\_screen(screenDefault, GTK\_STYLE\_PROVIDER(estilização),Prioridade da estilização); e por fim utilizamos o g\_object\_unref(provider) para evitar vazamentos de memória, para chama-lo inserimos na main antes de gtk\_widget\_show\_all(window); a chamada da função da seguinte forma:

carregar\_css();

gtk\_widget\_show\_all(window);

Agora que temos nossa janela pronta vamos começar a inserir eventos necessários no app farei uma listagem dos comandos necessários e cada um de seus respectivos eventos.

5.1 CRIANDO EVENTOS DE CONTROLE

5.1.1 CRIANDO EVENTOS DE BOTÃO

Para fazer a captura de eventos de click em botões basta seguir o exemplo abaixo:

GtkWidget \* Login= GTK\_WIDGET(gtk\_builder\_get\_object(builder, "NomeDoID")); g\_signal\_connect(Login, "clicked", G\_CALLBACK(on\_Abrir\_Chat), builder);

o seguinte trecho apresentado recebe o valor de um botão e conecta o sinal de click a uma função, sendo “ \*Login ” o botão, ‘Clicked’ o tipo do evento e G\_CALLBACK(nome\_da\_função\_de\_retorno),builder); a captura do clique e chamada da função por exemplo função:

void nome\_da\_função\_de\_retorno(GtkButton \*button, gpointer user\_data){}

então ao ser clicado no botão “Login” está função será acionada os parâmetros dentro da função uma delas deve sempre representar o tipo da função por exemplo por ser um GtkButton o parâmetro será o valor do botão pressionado ele ficará armazenado na função como button e gpointer é um retorno opcional definido no CALLBACK da função no caso o user\_data tem valor de builder, passado anteriormente.

5.1.2 PEGANDO VALORES DE GTKLABEL E INSERINDO VALORES

Caso um GtkLabel tenha sido inserido com um ID especifico, podemos manipula-lo dentro do código da seguinte forma.

Primeiro declaramos uma variável que contém o valor do label, por exemplo GtkLabel \*label = (GTK\_LABEL(gtk\_builder\_get\_object(builder, "nome\_do\_label"))); e então no ponteiro label temos o valor do GTKLABEL selecionado e agora se quisermos pegar o valor de dentro deste label utilizamos char \*valor\_do\_label = gtk\_label\_get\_text(label);

e então dentro de valor\_do\_label contém o texto atual do label, e para alterar o valor dentro do label utilizamos

gtk\_label\_set\_text(label,valor\_do\_texto);

quando vamos alterar o valor, podemos em valor\_do\_texto inserir um valor char ou inserir um texto dentro de “”, ambos funcionam corretamente, porém dentro de aspas ficará sempre um texto fixo já em uma variável ele pode ser alterado dinamicamente.

5.1.3 PEGANDO VALORES DE GTKENTRY, GTKENTRYSEARCH E UTILIZANDO SINAL CHANGE

Para conseguirmos pegar valores inseridos dentro de um GtkEntry precisamos primeiro pegar seu respectivo ID:

GtkEntry \*Pesquisar = (GTK\_ENTRY(gtk\_builder\_get\_object(builder, "ID do ENTRY")));

E então inserimos o seguinte comando retornando o valor do ENTRY

gchar \*Var = gtk\_entry\_get\_text(Pesquisar);

Agora dentro do ponteiro Var contem o valor do entry Pesquisar, para o GtkEntrySearch é necessário os mesmos passos, agora usando o sinal Changed e Search-Changed, para que o Sinal seja capturado sempre devemos deixa-los na main para evitar erros na hora da captura, o seguinte trecho de código captura os sinal Search-Changed e Changed

g\_signal\_connect(Pesquisar, "search-changed", G\_CALLBACK(funcao\_a\_ser\_disparada), retorno do pointer user\_data);

g\_signal\_connect(Pesquisar, "changed", G\_CALLBACK(funcao\_a\_ser\_disparada), retorno do pointer user\_data);

os seguintes trechos conectam um sinal de call-back na entry armazenada em Pesquisar e faz com que a função, seja disparada, é necessário termos armazenado o entry para inserir o sinal e então um exemplo de função é

void funcao\_a\_ser\_disparada (GtkEntry \*entry, gpointer user\_data){

gchar \*text = gtk\_entry\_get\_text(entry);

g\_print(“Valor da Entry: %s\n”, text);

}

A função a cada clique do usuário retorna no console o valor digitado pelo usuário assim é possível ser feito o teste para verificar se os sinais estão feitos da forma correta.

Para apagarmos ou inserir um valor dentro do Entry devemos utilizar:

GtkWidget \*Entry\_widget = GTK\_WIDGET(gtk\_builder\_get\_object(GTK\_BUILDER(user\_data), "ID do Entry"));

gtk\_entry\_set\_text(GTK\_ENTRY(Entry\_widget), "Valor a ser inserido");

Assim os valores presentes serão substituídos pelo valor presente após a virgula nos parâmetros, devemos nos atentar, pois diferente do momento de pegar o valor de um entry para armazena-lo, o valor será inserido no widget do entry.

5.1.4 PEGANDO VALOR DE CHECKBUTTON

O check button é um botão que irá retornar caso ativado True e caso desativado false, iremos capturar esses valores, podendo assim criar diversas codificações, caso o valor do Chech button seja true, seja false, o seguinte trecho de código pegamos o valor de um check button:

GtkWidget \*CheckButton= GTK\_WIDGET(gtk\_builder\_get\_object(builder, "Id\_do\_Check\_button"));

Com está propriedade armazenamos o valor do CheckButton no ponteiro CheckButton, e então para utilizarmos

g\_signal\_connect(CheckButton, "toggled", G\_CALLBACK(funcao\_de\_call\_back), builder);

Com este sinal a cada alteração no CheckButton retornará seu respectivo valor, ativado e desativado, função de exemplo :

Void função\_de\_call\_back(GtkToggleButton \*toggle\_button, gpointer user\_data)

{ gboolean botao\_valor = gtk\_toggle\_button\_get\_active(toggle\_button);

If (botao\_valor){ g\_print(“Foi Ativado”) }

Else { g\_print(“Está desativado ou foi desativado”)}

}

Sabendo que o valor retornado ao ser ativado é TRUE e valor ao estar desativado é FALSE, podemos fazer uma cadeia if, else para cada necessidade.

5.1.5 PEGANDO VALORES DE SWITCHS

O switch é muito semelhante ao Check button sendo que em caso do mesmo, ser ativado ele vai retornar TRUE e em caso desativado retornará FALSE o seguinte trecho de código faz a manipulação de um Switch:

GtkWidget \*Switch = GTK\_WIDGET(gtk\_builder\_get\_object(builder, "ID do Switch"));

Neste trecho armazenamos o valor do Switch no ponteiro Switch e então para capturarmos seu sinal usamos:

g\_signal\_connect(Switch, "state-set", G\_CALLBACK(funcao\_switch\_clicado), builder);

E agora neste trecho a cada ativação e desativação do usuário vai chamar a função “função\_switch\_clicado”, agora na função podemos adicionar parâmetros que receberam os valores no respectivo Switch:

gboolean função\_switch\_clicado(GtkSwitch \*switch\_button, gboolean state, gpointer user\_data)

{

if(state){

g\_print(“Switch Ativado”);

} else { g\_print(“Switch Desativado”); } }

5.2 EXIBINDO JANELAS DE MENSAGEM, ALTERANDO STACK ATUAL E UTILIZANDO GTKFILECHOOSERDIALOG

5.2.1 ALTERANDO STACK ATUAL

Para alterarmos o GtkStack presente dentro da página para alterarmos a exibição de uma janela para outra utilizamos o seguinte trecho de código

GtkStack \*stack = GTK\_STACK(gtk\_builder\_get\_object(builder, "ID do Stack"));

Com este trecho recuperamos o valor do Stack e inserimos dentro do ponteiro stack e então para fazermos sua respectiva troca de página utilizamos:

gtk\_stack\_set\_visible\_child\_name(stack, “Nome do Empacotamento”);

E então a janela será trocada para o Stack de Nome “Nome do Empacotamento” um exemplo de função que pode ser criada seria a seguinte:

void Clique\_troca\_de\_telas(GtkButton \*button, gpointer user\_data){

GtkStack \*stack = GTK\_STACK(gtk\_builder\_get\_object(user\_data, "ID do Stack")); gtk\_stack\_set\_visible\_child\_name(stack, “Nome do Empacotamento”);}

A função criada faz com que quando um botão é clicado ele faz a troca de uma widget presente no stack, nesta função considerei que o valor de user\_data teria como valor retornado builder caso builder esteja em uma variável global pode-se alterar o user\_data dentro do stack para builder.

5.2.2 EXIBINDO E ESCONDENDO JANELA DE MENSAGENS

Agora para exibirmos uma janela é um procedimento relativamente simples primeiro precisamos pegar o ID da janela de mensagem e armazena-lo:

GtkMessageDialog \*janela\_mensagem = GTK\_MESSAGE\_DIALOG(gtk\_builder\_get\_object(builder, "ID da janela de Mensagem"));

E então para inicia-la utilizamos o seguinte trecho:

gtk\_widget\_show\_all(GTK\_WIDGET(janela\_mensagem));

gtk\_dialog\_run(GTK\_DIALOG(janela\_mensagem));

gtk\_widget\_hide(GTK\_WIDGET(janela\_mensagem));

O gtk\_widget\_show\_all é responsável que faz com que mostramos a janela, já o gtk\_dialog\_run faz com que na janela seja necessário a desativação da mesma para utilizar as demais janelas ativadas, já o gtk\_widget\_hide faz com que ao ser clicado no X no topo da janela a mesma vai ser escondida assim retirando sua visualização até ser chamada novamente o gtk\_widget\_hide é extremamente necessário para o funcionamento da janela que em caso o usuário clique no X pode ocorrer bugs sem esse funcionamento ativado, caso você queira fazer um evento para esconder a

janela sem a necessidade de apertar o X, podemos fazer a utilização do seguinte trecho:

GtkMessageDialog \*janela\_mensagem = GTK\_MESSAGE\_DIALOG(gtk\_builder\_get\_object(builder,"ID da janela de Mensagem"));

GtkWidget \*tela\_da\_janela = GTK\_WIDGET(janela\_mensagem);

gtk\_widget\_hide(tela\_da\_janela);

Este trecho faz com que ao ser clicado em um botão por exemplo a função chamada pode esconder a respectiva janela presente.

5.2.3 UTILIZANDO GTKFILECHOOSSER

O filechoosser é muito importante na parte de download de arquivo em nosso projeto, pois ele armazena o caminho que o usuário especificou para efetuar o download de um respectivo arquivo, fazendo se tornar uma tarefa muito mais dinâmica o seguinte trecho inicia a janela:

GtkWidget \*file\_chooser\_dialog = GTK\_WIDGET(gtk\_builder\_get\_object(user\_data,"Id do file chooser "));

gtk\_widget\_show\_all(GTK\_WIDGET(file\_chooser\_dialog));

gtk\_dialog\_run(GTK\_DIALOG(file\_chooser\_dialog));

gtk\_widget\_hide(file\_chooser\_dialog);

Assim como as janelas anteriores o FileChooser deve ser aberto com o hide ao final para esconder a janela em caso usuário selecionar o X, agora o mais importante o botão for selecionado para ser caso o usuário escolher sua respectiva pasta deve conter na função de CallBack os seguintes dados:

GtkFileChooser \*chooser = GTK\_FILE\_CHOOSER(file\_chooser\_dialog);

char \*folder\_path = gtk\_file\_chooser\_get\_filename(chooser);

gtk\_widget\_hide(GTK\_WIDGET(chooser));

Desta maneira será armazenado em folder\_path o caminho selecionado pelo usuário e no fim fechamos a janela presente.

5.3 INSERINDO DADOS EM TABELA (GTK TREE VIEW) E UTILIZANDO SINAL ROW ACTIVATED

Para fazermos a inserção de dados que serão exibidos em uma tabela devemos preencher o liststore vinculado a mesma com dados e eles serão interpretados na tabela em suas respectivas colunas primeiro devemos pegar o componente liststore vinculado a tabela:

GtkListStore \*store1 = GTK\_LIST\_STORE(gtk\_builder\_get\_object(builder, "ID do list store"));

Neste trecho pegamos o valor de liststore e armazenamos em um ponteiro store1, após termos seu respectivo componente declaramos um iterador ele é necessário para modificação de dados de tabelas e textview, pois ele faz referência a posição numérica dos itens dentro dos componentes, podendo assim alterar e receber seus dados, o seguinte trecho declara um iterador:

GtkTreeIter iter;

Agora que temos um iterador para inserirmos dados dentro de uma liststore utilizamos o seguinte trecho:

gtk\_list\_store\_append(store1, &iter);

gtk\_list\_store\_set(store1, &iter, numerodacoluna,”valordacoluna” ,-1);

Primeiro o gtk\_list\_store\_append seta que iremos inserir um elemento a mais no store1 com o inter que retorna a posição dos itens do componente e então inserimos o valor dessa nova linha em gtk\_list\_store\_set com o numero da coluna e seu respectivo valor, é necessário inserirmos o -1 no final para sinalizar o final da lista de argumentos.

Utilizar listas podem ser inseridas consecutivamente em um comando while por isso sua utilidade é muito bem vinda, é importante lembrar que enquanto não sejam apagados o valor do liststore se mantem até o desligamento da aplicação, para limparmos um liststore se necessário utilizamos o seguinte trecho:

gtk\_list\_store\_clear(store1);

Este comando irá limpar todo o conteúdo presente no ponteiro store1.

Agora o sinal row\_activated faz com que ao usuário clicar duas vezes em uma linha da tabela ela ativará esse sinal podendo ser manipulado informações presentes na respectiva linha, a seguir o trecho para efetuar as declarações necessárias para uso:

GtkTreeView \*Tabela = GTK\_TREE\_VIEW(gtk\_builder\_get\_object(builder, "ID do tree view"));

g\_signal\_connect(Tabela, "row-activated", G\_CALLBACK(funcao\_da\_linha\_clicada), builder);

O comando anterior insere em Tabela o valor da Gtk\_tree\_view onde há o evento row e então conecta o sinal e então a cada clique ativa a função\_da\_linha\_clicada, um exemplo de uso da função seria a seguinte:

void função\_da\_linha\_clicada(GtkTreeView \*tree\_view, GtkTreePath \*path, GtkTreeViewColumn \*column, gpointer user\_data){

GtkTreeModel \*model = gtk\_tree\_view\_get\_model(tree\_view);

GtkTreeIter iter;

gchar \*nome;

if (gtk\_tree\_model\_get\_iter(model, &iter, path)){

if(g\_strcmp0(gtk\_tree\_view\_column\_get\_title(column), "Coluna 1") == 0){

gtk\_tree\_model\_get(model, &iter, 0, &nome, -1);

g\_print(“Voce clicou na coluna 1 que contém o seguinte valor: %s”,nome);

g\_free(nome);

}

else if(g\_strcmp0(gtk\_tree\_view\_column\_get\_title(column), "Coluna 2") == 0){

gtk\_tree\_model\_get(model, &iter, 1, &nome, -1);

g\_print(“Voce clicou na coluna 2 que contém o seguinte valor: %s”,nome);

g\_free(nome);

}}}

Nesta função pegamos o valor da coluna clicada pelo usuário e exibimos uma mensagem personalizada no console, primeiro nos parâmetros da função por padrão inserimos o ponteiro tree\_view que faz referencia a tabela que ativou a função, o ponteiro path que faz referencia a linha clicada pelo usuário, ponteiro column que faz referencia a coluna clicada e user data que tem o valor de builder no call-back desenvolvido, agora dentro da função declaramos o ponteiro model que recebe o model contido no parâmetro tree\_view, após isso declaramos um iterador e então dentro do primeiro if pegamos o iterador da linha path dentro da tree\_view model caso ocorra tudo certo o valor retornado será true e continuará a codificação e então dentro do próximo if utilizamos a função g\_strcmp0 da biblioteca glib que compara duas strings e então se o valor da coluna column for exatamente igual a cadeia de caracteres “Coluna 1” ele irá dentro da variável gchar nome armazenar o valor presente na linha e coluna clicada e o else if fará a mesma ação, porém em outra coluna e após isso liberamos a variável gchar nome para evitar vazamentos de memória.

5.4 RECUPERANDO VALORES DIGITADOS EM GTKTEXTVIEW

Para recuperarmos o valor de um textview devemos pegar seu valor:

GtkTextView \*Texto = GTK\_TEXT\_VIEW(gtk\_builder\_get\_object(user\_data, "Id do textview"));

Este trecho armazena o texteview no ponteiro texto, após esse procedimento devemos gerar um buffer com base no textView presente no ponteiro texto:

GtkTextBuffer \*buffer = gtk\_text\_view\_get\_buffer(Texto);

Agora com o buffer criado devemos criar dois iteradores que serão um para o inicio e outro para o fim dos valores digitados no text view:

GtkTextIter start, end;

Após ambos iteradores criados devemos fazer com base no buffer com que os iteradores recebam os valores iniciais e finais digitados:

gtk\_text\_buffer\_get\_start\_iter(buffer, &start);

gtk\_text\_buffer\_get\_end\_iter(buffer, &end);

Então declaramos um gchar onde será armazenado o valor textual do TextView e o mesmo poderá ser manipulado:

gchar \*texto = gtk\_text\_buffer\_get\_text(buffer, &start, &end, FALSE);

g\_free(texto);

Devemos sempre ao final liberar o valor de um gchar com o g\_free(nome do gchar), então dentro do ponteiro texto foi armazenado o valor do gtktextview sendo que os parâmetros da função gtk\_text\_buffer\_get\_text deve sempre receber por parâmetro o buffer após ele o iterador inicial e após o final, o valor FALSE é para caso haja caracteres escondidos, porém por padrão deixaremos como FALSE.

Para apagarmos os valores do text View devemos utilizar o seguinte trecho:

gtk\_text\_buffer\_set\_text(buffer, "", -1);

O valor presente no textview será substituído pelo presente em “”.

5.5 UTILIZANDO MYSQL CONNECTOR

O mysql conector é necessário para fazermos os comandos que serão direcionados ao banco de dados para darmos inicio precisamos criar de preferencia uma função para o retorno da conexão para utilizarmos no código e está conexão fará com que nossos comando sejam enviados para o banco de dados, a função de conexão é a seguinte:

MYSQL \*\_stdcall obterConexao()

{ char \*servidor = "127.0.0.1";

char \*usuario = "root";

char \*senha = "";

char \*nomeBanco = "projeto\_ambiental";

MYSQL \*conexao = mysql\_init(NULL);

if(!mysql\_real\_connect(conexao, servidor, usuario,senha, nomeBanco, 3306,NULL, 0))

{g\_print("Não foi Possivel conectar ao Banco de Dados ") ;

mysql\_close(conexao);}

else{ if (mysql\_set\_character\_set(conexao, "utf8")){

fprintf(stderr, "Erro ao configurar UTF-8: %s\n", mysql\_error(conexao));}

return conexao;}}

Neste trecho utilizamos diversas funções do mysql.h que fazer o início da conexão com o banco, primeiro chamamos MYSQL com o ponteiro \_stdcall e então nomeamos a sua função como obterConexao(), então caso criarmos uma conexão a função a ser chamada é obterConexao(), no inicio da função declaramos 4 ponteiros char necessários no momento de iniciar a conexão, primeiro o ponteiro servidor, dentro dele será armazenado o local do servidor mysql instalado, o banco de dados utilizado por mim foi o que vem incluso ao xampp então o servidor por padrão seria [localhost](http://localhost/) ou o utilizado por mim “127.0.0.1” utilizando o diretamente o endereço de ip, agora em usuário utilizei o que vem por padrão em qualquer servidor, o usuário “root”, e o ponteiro senha, é a senha do usuário root, no meu caso não há então recebe apenas um valor vazio “” e em nomeBanco ele deve receber o nome do banco que iremos utilizar no meu caso é o “projeto\_ambiental”, dentro do installer haverá os comandos para criação do respectivo banco criado, e após isso chamamos um ponteiro com MYSQL nele ficará armazenado a conexão em caso de sucesso e então inserimos o valor de mysql\_init(NULL) onde há o mysql\_init é onde receberá os valores para iniciar conexão a principio inserimos o valor NULL para caso de erro retornar false no if a seguir e então iniciamos o if com mysql\_real\_connection onde temos os seguintes parâmetros (ponteiro com mysql\_init,servidor referente, usuário, senha, nomedoBanco, porta do servidor do banco, socket\_unix que no meu caso como não há o valor é NULL, e 0 no último parâmetro por padrão) e então no início incluímos um ! para o valor caso seja TRUE, ou seja conexão deu certo retorne false e não entre no IF e então no ELSE em caso conexão seja sucesso incluímos um ultimo if com valor de mysql\_set\_character\_set() esse caso é para a conexão enviar dados formatados em diferentes formatos no meu caso os parâmetros é a conexão no inicio e o segundo parâmetro é o estilo de formatação dos dados que é utf8 para formatação brasileira com suporte a caracteres especiais dentro do if em caso de erro inserimos uma mensagem e então retornamos conexao para o uso dos dados já configurados.

Agora com a função criada podemos fazer as chamadas necessárias para fazer as consultas e inserção de dados no banco, por padrão temos que fazer a declarações de alguns ponteiros para fazermos a manipulação de dados caso fizermos uma Requisição SELECT análise o exemplo abaixo:

MYSQL \*conexao=obterConexao();

MYSQL\_RES \*Valor\_Select = NULL;

MYSQL\_ROW = COLUNA

Char consulta[250];

snprint(consulta,sizeof(consulta),”SELECT \* FROM nome\_tabela”);

If(mysql\_query(conexao, consulta)==0)

Valor\_Select = mysql\_store\_result(conexao);

If(Valor\_Select != NULL && mysql\_num\_row > 0)

{ while((row = mysql\_fetch\_row(Valor\_Select)) != NULL)

{ g\_print(“%s”,row[0]);}

mysql\_free\_result(Valor\_Select);

}

mysql\_close(conexao)

Agora explicarei cada etapa do código, primeiro declaramos dentro de um ponteiro MYSQL o valor do retorno da função cbterConexao está variável contém a conexão com o banco, após isso declaramos um ponteiro em MYSQL\_RES ele fará com que consigamos armazenar o valor retornado pelo banco para manipulação e por ultimo declaramos um ponteiro MYSQL\_ROW nele será armazenado as informações coluna por coluna, podendo então pegar cada valor separadamente e então para efetuarmos a pesquisa declaramos uma cadeia de caracteres consulta e então utilizamos o snprinf do include stdio.h para inserir dentro dessa cadeia os dados de pesquisa, dependendo do tipo de informação inserida diferentes consultas e adição de dados para o banco podem ser feitas, porem é necessário algum conhecimento sobre linguagem sql para efetuar essas criações, após prepararmos a consulta devemos utilizar o mysql\_query com dois parâmetros obrigatórios o primeiro é a conexão com o banco de dados e o outro é o comando sql criado é sempre bom inserir esses comandos dentro de um if para tratativas em caso de erro, dentro do if fiz a comparação com 0, pois caso não ocorra nenhum erro o retorno será 0 e assim dentro do if fazemos a agregação do retornado ao Valor\_Select utilizando mysql\_store\_result e então fazemos mais uma tratativa, pois Valor\_Select, pode não estar com nenhum valor ou pelo erro na consulta o mesmo pode ainda estar com o valor padrão NULL definido, então para essas verificações utilizamos um if onde Valor\_Select tem que ser diferente de NULL e também o numero de resultados, representado por my\_sql\_num\_row deve ser maior que 0 e assim iniciamos a utilização dos dados, para isso usamos um while que como seus parâmetros armazena em row cada coluna de Valor\_Select enquanto os valores forem diferente de NULL e dentro do while printamos no console o valor da primeira coluna com o valor row[0] devemos sempre levar em conta a quantidade de colunas presentes que fizemos a pesquisa, por exemplo em caso de 2 colunas devemos inserir row[0] para a primeira e row[1] para a segunda e assim suscetivelmente,OBS: o comando while é processado até que todas as linhas dos itens retornados acabe e para acabar liberamos o valor do ponteiro Valor\_Select para caso seja necessário ser reutilizado na função e fechamos a conexão aberta em conexao.

E assim pegamos os valores retornados por um SELECT caso utilize comandos para inserção de dados como o INSERT o código pode ser finalizado após a query, pois não a dados a serem retornados.

5.6 EFETUANDO REQUISIÇÕES HTTP COM CURL

Requisições com o curl variam entre os tipos, onde cada um deve receber um tratamento diferente, irei citar requisições utilizando o método GET e POST.

Primeiro em curl vamos iniciar pelas necessidades básicas onde ambas requisições necessitam destas configurações:

CURL \*curl;

CURLcode resposta;

struct MemoryStruct chunk;

chunk.memory = malloc(1);

chunk.size = 0;

curl = curl\_easy\_init();

if (curl){

curl\_easy\_setopt(curl, CURLOPT\_URL, “url/para/o/servidor”);

curl\_easy\_setopt(curl, CURLOPT\_WRITEFUNCTION, WriteMemoryCallback);

curl\_easy\_setopt(curl, CURLOPT\_WRITEDATA, (void \*)&chunk);

resposta = curl\_easy\_perform(curl);

if (resposta != CURLE\_OK)

{ fprintf(stderr, "Erro no cURL: %s\n", curl\_easy\_strerror(resposta)); } curl\_easy\_cleanup(curl);

free(chunk.memory);

}

Este são os comandos básicos em curl para se enviar uma requisição e armazena-las para caso haja algum retorno do servidor, de inicio declaramos o ponteiro curl nele que será feito as configurações e envio da requisição, resposta é onde vai ser retornado o status da requisição, agora antes de continuarmos deve enfatizar o uso do struct, o struct será necessário para a alocação dinâmica de memória para a resposta vinda do servidor, o struct será o seguinte:

struct MemoryStruct

{char \*memory;

size\_t size;};

Neste struct contém memory onde será armazenado os dados e size onde ficará o tamanho de alocação necessário.

Então voltando ao curl chamamos o struct e inserimos em chunk , após isso definimos memory de chunk com tamanho de 1 byte ou seja vazio e size igual 0, agora iniciamos a configuração da requisição com curl\_easy\_init(); utilizamos o if para caso o retorno de curl seja FALSE houve erro no inicio da configuração, dentro do if então utilizamos curl\_easy\_setopt para inserir em curl os detalhes da requisição, o primeiro easy\_setopt insere o URL onde será feito a requisição, já o segundo devemos criar uma função para alocação dinâmica do retorno de dados do servidor para o struct, o terceiro complementa o segundo sendo que ele faz em (void \*)&chunk o envio em userp o endereço de memória tratado na função necessária abaixo:

size\_t WriteMemoryCallback(void \*contents, size\_t size, size\_t nmemb, void \*userp)

{size\_t realsize = size \* nmemb;

struct MemoryStruct \*mem = (struct MemoryStruct \*)userp;

char \*ptr = realloc(mem->memory, mem->size + realsize + 1);

if (ptr == NULL){

printf("Não foi possível alocar memória.\n");

free(mem->memory);

return 0;}

mem->memory = ptr;

memcpy(&(mem->memory[mem->size]), contents, realsize);

mem->size += realsize;

mem->memory[mem->size] = 0; // Null-terminando o buffer

return realsize;}

Essa função aloca mais memória dentro do struct para novos dados vindos do servidor sem alteração de dados já presentes, fazendo com que size se torne o tamanho do total de dados armazenados e memory o valor dos dados recebidos e armazenados, os parâmetros da função contem contents que representa os dados recebidos da requisição, size os valores de cada unidade recebida pela requisição, nmemb o número de unidades presentes na requisição e userp que apontará em CURLOPT\_WRITEDATA para o struct onde os dados foram enviados que no nosso caso faz referência ao endereço de memória de chunk, então agora dentro da função fazemos o cálculo dos dados de size \* nmemb e esse valor representa o valor total dos dados recebidos e armazenamos em realsize, depois criamos o struct mem que é igualado ao struct chunk retornado por userp, após isso usamos realloc para alocar mais memória em memory e ajustamos mem->size para receber esses novos dados, levando em conta os já existente, somando a realsize, e +1 ao final para dar espaço para um terminador null, então armazenando esses valores em ptr depois para caso não seja possível a alocação fazems um if para caso o valor retornado por ptr seja null, caso não seja null, fazemos com que memory receba a nova memória alocada, então fazemos a cópia dos dados da requisição presentes em contents e seu respectivo tamanho para memory levando em conta os dados já armazenados em memory[size] alocando seu conteúdo ao final do conteúdo já presente, após a alocação fazemos com que size receba seu valor somado a de realsize para o novo tamanho e finalizamos com mem->memory[mem->size] recebendo o seu último valor como null. Agora Continuando dentro do curl após inserirmos os códigos para alocação de memória dinâmica iniciamos dentro de resposta a requisição com curl\_easy\_perform sendo como parâmetro curl com todas as devidas configurações e tratativas de dados, em resposta será armazenado o status da requisição em caso falha ou sucesso, fazemos essa tratativa no if que vem a seguir que em caso resposta seja diferente de CURL\_OK inserimos uma mensagem de erro personalizada com o erro em questão, então fazemos a limpeza do ponteiro curl que recebeu a requisição, e ao final liberamos a memória alocada em chunk.memory.

5.6.1 EFETUANDO REQUISIÇÃO GET

Para realizarmos requisições via GET o método é bem simples sendo no momento de inserir a url devemos ao final inserir o nome e valor necessário, por exemplo: <http://caminhodoservidor/arquivos?download> =nomedoarquivo e então ao servidor receber essa requisição ele pode retornar o arquivo para download na maquina do usuário ou um erro caso nomedoarquivo seja inexistente.

5.6.2 EFETUANDO REQUISIÇÃO POST

Agora para efetuarmos requisição POST é um pouco diferente sendo necessário adicionar dois novos campos junto de curl\_easy\_setopt sendo eles:

curl\_easy\_setopt(curl, CURLOPT\_POST, 1L);

curl\_easy\_setopt(curl, CURLOPT\_POSTFIELDS,"{\"download\": \"nomedoarquivo\"}");

O primeiro easy\_setopt seleciona o método POST como requisição, já o segundo seleciona os dados que serão enviados para a tratativa no servidor, devemos inserir de preferencia um formato JSON para o recebimento no servidor, este exemplo tem o mesmo efeito que o método GET.

5.7 EFETUANDO CRIPTOGRAFIA COM LIBSODIUM

Para efetuarmos a criação de um hash de dados como a senha do usuário utilizamos o seguinte trecho de código

char criptografia[crypto\_pwhash\_STRBYTES];

char Senha[] = “123456”;

if(sodium\_init() < 0){

g\_print(“Erro ao ativar sodium”);

return 1; }

if(crypto\_pwhash\_str(criptografia, Senha, strlen(Senha),

crypto\_pwhash\_OPSLIMIT\_INTERACTIVE,

crypto\_pwhash\_MEMLIMIT\_INTERACTIVE) != 0)

{g\_print("Erro ao gerar a criptografia\n");

return 1;}

Assim um hash da Senha é gerado e armazenado na variável criptografia. Este hash é seguro e adequado para armazenar senhas sem manter a senha original, para verificarmos no momento de login se a senha digitada pelo usuário está correta comparamos o hash já criado com um novo hash que compara ambos e verifique se está corretamente digitado, para isso pegamos o hash anteriormente gerado e o valor digitado pelo usuário, por exemplo:

char senha\_digitada[] = “123456789”;

if (crypto\_pwhash\_str\_verify(criptografia, senha\_digitada, strlen(senha\_digitada)) != 0){ g\_print(“Senha incorreta digitada”); }

else{ g\_print(“Senha correta digitada”); }

Este trecho compara o hash anterior e a senha digitada atual então o if faz o hash da senha\_digitada e compara com o criado anteriormente e no caso como senha\_digitada não é igual a Senha então ocorre o print da senha incorreta digitada e assim sucetivamente.

5.8 CAPTURANDO TEMPO PARA ARMAZENAMENTO DE DADOS

Com o include time.h podemos capturar o tempo em que os arquivos foram digitados pelo usuário o seguinte trecho armazena em formato DATETIME para banco de dados, para armazenamos dados com a data atual:

char data\_hora[20];

time\_t tempo\_atual;

time(&tempo\_atual);

struct tm \*data\_hora\_local = localtime(&tempo\_atual);

sprintf(data\_hora, "%04d-%02d-%02d %02d:%02d:%02d",

data\_hora\_local->tm\_year + 1900,

data\_hora\_local->tm\_mon + 1,

data\_hora\_local->tm\_mday,

data\_hora\_local->tm\_hour,

data\_hora\_local->tm\_min,

data\_hora\_local->tm\_sec);

A cadeia de data e hora é onde será armazenado o tempo atual, tempo\_atual por ser um time\_t pode se armazenar os dados de datas que vem a seguir, então utilizamos a função time(&time\_atual) armazena o tempo atual em segundos, depois chamamos um struct pré definido do include time onde serão feitos os cálculos de ano, mês, dia, hora e segundos e então poderão ser utilizados, então utilizamos o snprinf onde data\_hora receberá os valores dentro do struct data\_hora\_local onde ano recebe tm\_year, é necessário o cálculo de +1900, pois por padrão do struct sempre devemos somar a 1900, depois inserimos em mês tm\_mon + 1, pois o mês se inicia em 0 e para ficar padrão de 12 meses somamos 1 para representar de 1 até 12, e então pegamos dia em tm\_mday, hora em tm\_hour, minuto em tm\_min e segundos em tm\_sec, assim data e hora de snprintf receberá os valores de data e hora formatados.

5.9 INSERINDO E UTILIZANDO CJSON

O cJSON por ser um código C para implementarmos em nosso include devemos inserir o caminho de cJSON.h, para isso fazemos o include

#include “.\cJSON/cJSON.h”

Este trecho de include faz uma busca na pasta atual e procura pela pasta cJSON e dentro dela procura o arquivo de cabeçalho cJSON.h que faz referencia a cJSON.c assim o cJSON será adicionado, tomando de referencia o retorno da API ViaCep para um preenchimento automático de dados de cadastro, com retorno json vamos formata-lo para armazenar os dados em um variável vamos levar em consideração que a resposta foi armazenada em chunk.memory após uma requisição GET via curl:

cJSON \*json = cJSON\_Parse(chunk.memory);

O trecho anterior pega o valor json recebido em chunk.memory e converte para um valor cJSON valido e o mesmo é armazenado no ponteiro json, após isso no seguinte trecho:

if (json == NULL)

{fprintf(stderr, "Erro ao fazer parsing do JSON: %s\n", cJSON\_GetErrorPtr());

return;}

É feita uma tratativa para caso o valor de chunk.memory inserido no ponteiro json não seja valido:

const cJSON \*bairro = cJSON\_GetObjectItemCaseSensitive(json, "bairro");

if (bairro != NULL)

{ g\_print("%s", bairro->valuestring); }

else { g\_print("Bairro não encontrado!\n"); }

Neste trecho declaramos uma constante que pegará o objeto bairro nos dados json e irá inserir seu valor no ponteiro bairro, lembrando que por ser um CaseSensitive ele irá diferencial maiúsculas de minusculas, para detalhes de como é feito o retorno dos dados Json verifique no site <https://viacep.com.br>, e ao final ele pega o valor de bairro e printa no console, caso seu valor seja null ele retorna else com a mensagem de erro personalizada, sempre que precisarmos acessar seu conteúdo é necessário a utilização de ->valuestring para sua conversão legível, mas caso seja um valor inteiro utilize valueint e caso seja um valor flutuante utilize valuedouble.

6. FUNCIONAMENTO DO CÓDIGO CRIADO

Com todas as funcionalidades passo a passo vou fazer o código de forma teórica de como foi feito cada etapa, com as funcionalidades necessárias e então dentro da pasta zip conterá o instaler do aplicativo já feito e caso não seja possível devido o seu tamanho deixarei o código e suas respectivas pastas de testes criadas.

De inicio criamos uma pagina principal com explicações referentes aos projetos e o mesmo há botões para efetuar o login, ao ser clicado o Stack é alterado para login de usuário que contém um botão para retorno a página principal e campo de email e senha de usuário para login, é previamente criado o banco e nele gerado um usuário administrador via sql dentro do arquivo comando.sql há o respectivo usuário e senha que deverá ser inserida no banco, utilizando esse usuário logamos no stack como admin neste Stack o usuário tem direito de verificar relatórios das empresas cadastradas podendo efetuar o download, valores total contribuídos, janela de mensagem para relatório detalhado que pode ser filtrado por data, janela de mensagem para criação de usuário, janela de mensagem para criação de relatório, ainda uma parte de visualização de chamados recebidos tela de bate papo e envio de mensagens importantes, porém estes não serão citados devido sair da demanda principal, ao clicar em CRIAR USUARIO a janela de mensagem com Entrys para criação dos dados de cadastro de usuário empresa e adms é aberta, então pode ser inserido os dados necessários para criação do respectivo usuário e o mesmo será armazenado no banco de dados, dentro dos entrys de criação de usuário empresa ao ser digitado o CEP é feito uma requisição ao viaCep e os valores restantes são automaticamente preenchidos, ao sair da janela de mensagem e clicando em criar relatório a janela de criação de relatório é aberta primeiro nela selecionamos a empresa que terá seu relatório e dentro do relatório será inserido a quantidade de resíduos tratados e o valor de custo desses resíduos, então poderá ser criado um relatório com nome padrão ou há a possibilidade de um nome especificado em Nome do Arquivo ao finalizar o respectivo documento, irá ser enviado para nosso servidor para ser armazenado o arquivo para download e certos dados que serão utilizados com frequência no layout ao sair da janela já será visível na tabela de relatórios os relatórios criados para a respectiva empresa e o download do arquivo em formatos xls, csv e txt e também aparecerá em valor total contribuído o valor de reais de cada relatório e em relatório detalhado ao ser clicado o relatório, também estará visível para visualização ao clicar em um icone de Download, vai ser aberto a janela para selecionar o local de download e então o arquivo será baixado no caminho especificado, então pode ser feito o logout ao fazermos o login como usuário da empresa criado os seus respectivos relatórios estarão presentes para download e verificação, a criação dos arquivos foi feito utilizando FILE manipulador de arquivos presente no include stdlib.

7. CONCLUSÃO

Após a extensa explicação de como é feito o funcionamento de cada etapa e itens utilizados no código, configuração de IDE e criação de interface pode ser criada uma aplicação útil para suprir a demanda da empresa, no código final haverá o código criado por mim onde tem toda a aplicação desenvolvida e suas respectivas funcionalidades caso, seja possível no arquivo zip será disponibilizado um installer onde preparara todo o ambiente em que precisamos para rodar a aplicação sendo apenas necessário o download do xampp para o lado do servidor, a aplicação tem diversas funcionalidades interessantes além das obrigatórias então adoraria um feedback referente a mesma.

Ao seguir toda explicação será possível não só conseguir entender o código já criado, mas criar novas funcionalidades de interface para uma infinidade de necessidades, por gtk ser uma linguagem muito maleável é possível desenvolvermos uma aplicação com multi plataformas de compatibilidade assim tendo uma escalabilidade maior para projetos.

Agradeço desde já e espero que a criação desse algoritmo e suas devidas explicações possam suprir a demanda necessária na startup de soluções ambientais, foi um projeto onde a codificação foi criada em um total de 2 semanas e 5 dias e a documentação levou cerca de uma semana para a criação desta documentação, por ser meu primeiro contato com a linguagem C e todas essas inclusões no código demandou uma grande e extensa pesquisa a seguir serão inseridas todas as respectivas referências onde precisei estudar para aprimorar meu embasamento e desenvolvimento nas demandas ressaltadas.

REFERENCIAS:

<https://www.treinaweb.com.br/blog/criando-interfaces-graficas-no-c-com-gtk#google_vignette>

<https://www.udemy.com/course/interface-grafica-para-apps-feitas-em-c-com-gtk-e-glade/?couponCode=KEEPLEARNING>

<https://www.youtube.com/watch?v=eLYZzIV-C64>

<https://curl.se/libcurl/c/>

<https://github.com/DaveGamble/cJSON>

<https://petbcc.ufscar.br/time/>