



PROJETO DE OFICINA DE INTEGRAÇÃO

Desenvolvimento de software para o empréstimo de livros para a biblioteca da UTFPR - Campus Pato Branco

Acadêmicos: Gabriel Henrique Meurer RA: 1917390
Luiza Stringhini Linhares RA: 1978896

1. Introdução

Com o objetivo de otimizar o empréstimo de livros em uma biblioteca, o software Bibliotec Digital foi desenvolvido. De modo geral, o projeto pode ser dividido em dois, o Backend e o Frontend, para o Backend foi desenvolvido um banco de dados no software Postgre em que contém todas as informações da biblioteca, dentre elas informações de livros, alunos, data de devolução, etc. Para o Frontend foi desenvolvida a interface gráfica em Python Kivy juntamente com a linguagem Kivy que facilita a manipulação dos layouts. Neste documento será apresentado com detalhes como foi o processo de desenvolvimento do software.

2. Desenvolvimento

Primeiramente, foi realizado o desenvolvimento do banco de dados onde seria feito o armazenamento e manipulação dos livros, alunos, funcionários, e os empréstimos realizados. Para isto, utilizamos do pgAdmin4, um software que utiliza a linguagem PostgreSQL para a execução do banco de dados. Em seguida, foram criadas as tabelas dos atributos mencionados acima, e, através do Mockaroo, foram gerados dados aleatórios que pudessem ser utilizados para preencher as tabelas livro e aluno.

```
-- Tabela aluno
create table if not exists public.aluno (
  RA bigint not null check (RA > 0) UNIQUE,
  nome varchar(50) not null,
  senha real not null,
  email varchar(50),
  curso varchar(50),
  multa real default 0,
```

```

    emprestados jsonb default null,

    primary key (RA)
);

-- Tabela Funcionarios
create table if not exists public.funcionario (
    matricula bigint not null check (matricula > 0) UNIQUE,
    nome varchar(50) not null,
    senha real not null,
    email varchar(50),

    primary key (matricula)
);

-- Tabela livros
create table if not exists public.livro (
    id bigint not null default nextval('livro_id_sequence') check (id
> 0),
    titulo varchar (150),
    autor varchar(50),
    categoria varchar(15),
    disponibilidade boolean default true,

    primary key(id)
);

-- Tabela emprestimos
create table if not exists public.emprestimo (
    livro_id bigint not null UNIQUE,
    aluno_RA bigint not null,
    horario timestamp not null,
    devolucao date not null,

    foreign key(livro_id) references public.livro (id),
    foreign key(aluno_RA) references public.aluno (RA)
);

```

Em seguida, foram feitas determinadas funções e triggers para garantir o funcionamento correto do banco, como, por exemplo: calculo_multa, emprestimo_livro e devolucao_livro, mostradas no script abaixo:

```

-- Função para calculo de multa

create or replace function calculo_multa (devolucao_aux date)
returns real as $$
begin
    return CURRENT_DATE - devolucao_aux;

```

```

end
$$ language plpgsql;

-- Trigger Para Emprestimos de Livros
CREATE OR REPLACE FUNCTION emprestimo_livro()
RETURNS trigger AS $empr$
DECLARE
    aluno_RA_aux bigint;
    livro_id_aux bigint;
    devolucao_aux date;
    emprestados_aux jsonb;
    disponibilidade_aux boolean;
    multa_aux real;
BEGIN
    select RA into aluno_RA_aux from public.aluno where
public.aluno.RA = new.aluno_RA;
    select id into livro_id_aux from public.livro where
public.livro.id = new.livro_id;
    select multa into multa_aux from public.aluno where
public.aluno.RA = new.aluno_RA;
    select devolucao into devolucao_aux from public.emprestimo where
devolucao = new.devolucao;
    select disponibilidade into disponibilidade_aux from public.livro
where public.livro.id = new.livro_id;
    select emprestados into emprestados_aux from public.aluno where
public.aluno.RA = new.aluno_RA;

    if disponibilidade_aux is true
    then
        if multa_aux > 0
        then
            raise exception 'Aluno possui multa de % Reais a pagar',
multa_aux
            using hint = 'Pague antes de adquirir outros livros';
        else
            if emprestados_aux is not null
            then
                emprestados_aux := emprestados_aux::jsonb || (
                    '{
                        "' || livro_id_aux || '" :
                        {
                            "devolucao" :"' || devolucao_aux || '"
                        }
                    }'
                )::jsonb;
            else
                emprestados_aux := (
                    '{
                        "' || livro_id_aux || '" :
                        {
                            "devolucao" :"' || devolucao_aux || '"
                        }
                    }'
                )::jsonb;
            end if;
            update public.aluno set emprestados = emprestados_aux
where RA = aluno_RA_aux;

```

```

        update public.livro set disponibilidade = false where id
= livro_id_aux;
    end if;
end if;
refresh materialized view emprestimo_livros;
RETURN NEW;
END;
$empr$ language plpgsql;

CREATE TRIGGER insert_emprestimo
AFTER INSERT ON emprestimo
FOR EACH ROW
EXECUTE PROCEDURE emprestimo_livro();

-- Trigger para devolucao de livros
CREATE OR REPLACE FUNCTION devolucao_livro()
RETURNS TRIGGER AS $dvlc$
DECLARE
    livro_id_aux bigint;
    aluno_RA_aux bigint;
    multa_aux real;
    emprestados_aux jsonb;
    devolucao_aux date;
BEGIN
    select RA into aluno_RA_aux from public.aluno where
public.aluno.RA = old.aluno_RA;
    select id into livro_id_aux from public.livro where
public.livro.id = old.livro_id;
    select emprestados into emprestados_aux from public.aluno where
public.aluno.RA = aluno_RA_aux;
    select multa into multa_aux from public.aluno where
public.aluno.RA = aluno_RA_aux;
    select devolucao into devolucao_aux from public.emprestimo where
old.livro_id = livro_id_aux;

    UPDATE public.aluno SET emprestados = emprestados_aux - CAST
(livro_id_aux AS text) where public.aluno.RA = aluno_RA_aux;
    UPDATE public.livro SET disponibilidade = true where
public.livro.id = livro_id_aux;
    if calculo_multa(old.devolucao) > 0
    then
        UPDATE public.aluno SET multa = multa_aux +
calculo_multa(old.devolucao) where public.aluno.RA = aluno_RA_aux;
    end if;
    refresh materialized view emprestimo_livros;
    return null;
END;
$dvlc$ language plpgsql;

CREATE TRIGGER delete_emprestimo
AFTER DELETE ON emprestimo
FOR EACH ROW
EXECUTE PROCEDURE devolucao_livro();

```

Para então finalizar o banco, foram criadas as materialized view para facilitar a visualização de determinados dados do banco, assim como as tabelas de auditoria para manter o controle de tudo que é inserido, deletado ou atualizado no banco.

```
-- Materialized view para emprestimos

create materialized view emprestimo_livros (id_livro, titulo, RA_aluno,
nome, horario, devolucao)
as select
    emprestimo.livro_id,
    livro.titulo,
    emprestimo.aluno_RA,
    aluno.nome,
    emprestimo.horario,
    emprestimo.devolucao
from public.emprestimo, public.livro, public.aluno
where public.emprestimo.livro_id = public.livro.id and
public.emprestimo.aluno_RA = public.aluno.RA order by horario desc;

-- Auditoria emprestimo
create table if not exists audit.emprestimo (
    usuario varchar,
    data timestamp,
    operacao char,
    livro_id bigint not null,
    aluno_RA bigint not null,
    horario timestamp not null,
    devolucao date not null
);

-- Auditoria livro
create table if not exists audit.livro (
    usuario varchar,
    data timestamp,
    operacao char,
    id bigint not null,
    titulo varchar (150),
    autor varchar(50),
    categoria varchar(15),
    disponibilidade boolean
);

-- Auditoria da tabela emprestimo
create or replace function audit_emprestimo()
returns trigger as $audit_emp$
begin
    if (TG_OP = 'INSERT')
    then
        insert into audit.emprestimo select user, now(), 'I', new.*;
        return new;
    elsif (TG_OP = 'DELETE')
```

```

        then
            insert into audit.emprestimo select user, now(), 'D', old.*;
            return new;
        elsif (TG_OP = 'UPDATE')
        then
            insert into audit.emprestimo select user, now(), 'U', new.*;
            return new;
        end if;

        return null;
    end;
$audit_emp$ language plpgsql;

create trigger audit_emp
after insert or update or delete on public.emprestimo
for each row execute procedure audit_emprestimo();

-- Auditoria da tabela Livro
create or replace function audit_livro()
returns trigger as $audit_lvr$
begin
    if (TG_OP = 'INSERT')
    then
        insert into audit.livro select user, now(), 'I', new.*;
        return new;
    elsif (TG_OP = 'DELETE')
    then
        insert into audit.livro select user, now(), 'D', old.*;
        return new;
    elsif (TG_OP = 'UPDATE')
    then
        insert into audit.livro select user, now(), 'U', new.*;
        return new;
    end if;

    return null;
end;
$audit_lvr$ language plpgsql;

create trigger audit_livro
after insert or update or delete on public.livro
for each row execute procedure audit_livro();

```

Após o desenvolvimento do banco de dados, foi feito o desenvolvimento do leitor de código de barras. Para isso, utilizamos uma biblioteca em python chamada pyzbar, que possui uma função que decodifica os dados inseridos no código de barras assim que o mesmo for apresentado na câmera do computador.

Na etapa de execução do Frontend primeiramente foi feito um protótipo navegável no software web Figma, como mostrado na Figura 1. Tendo isso como base, foi escolhida uma biblioteca do Python chamada Kivy para desenvolver a interface. A biblioteca Kivy foi optada pelos fatores de que, além de ser uma biblioteca, também possui uma linguagem própria que se integra ao Python e que facilita a manipulação dos layouts a serem codificados.

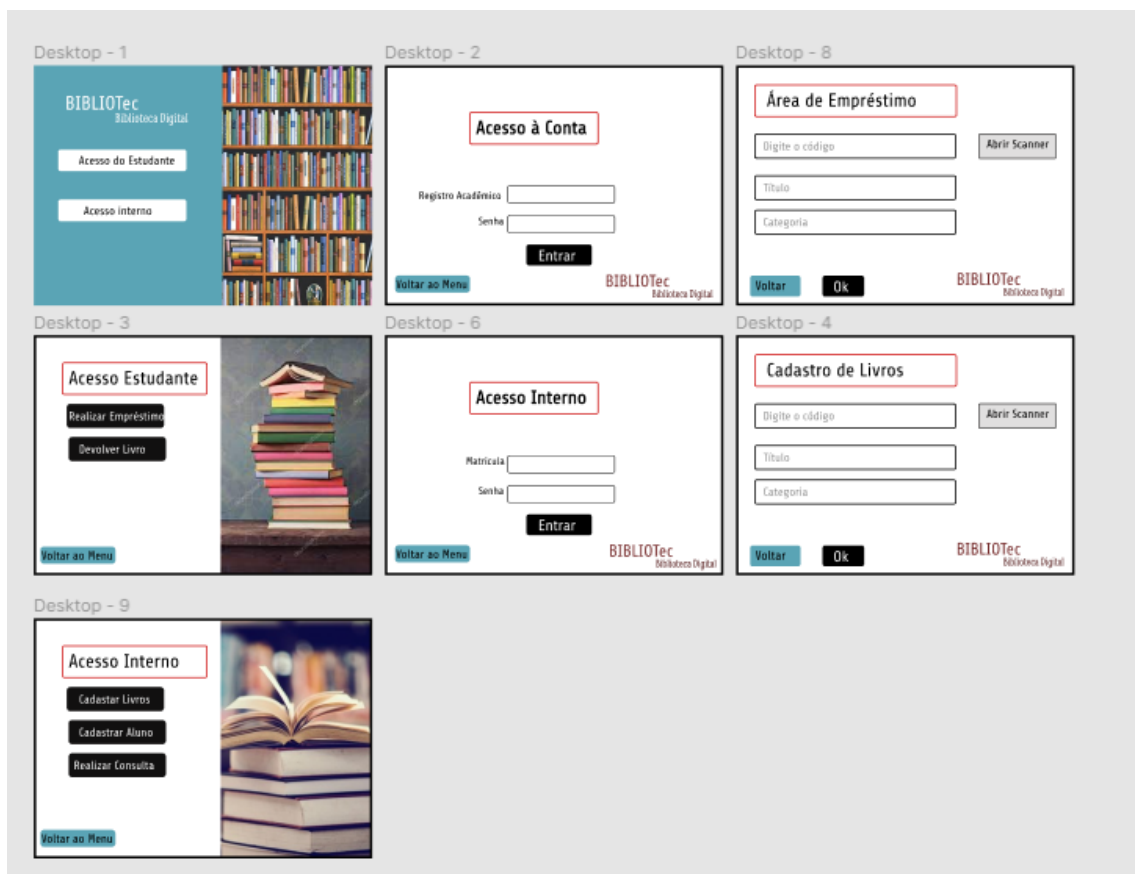


Figura 1 - Protótipo da Interface desenvolvido no Figma.
Fonte: Os Autores

Para uma melhor compreensão de como funciona a linguagem Kivy e o Python trouxemos uma exemplificação da tela Empréstimo onde é selecionado se o aluno deseja emprestar ou devolver um livro.

```
class Empréstimo1(Screen):
    pass
class Bibliotec(App):
    def build(self):
        sm = ScreenManager()
        sm.add_widget(Empréstimo1(name='emprestar'))
```

Como pode-se ver acima no arquivo Python é apenas criada uma classe com o atributo do tipo tela, e na classe construtora da aplicação é adicionada através do gerenciador de telas. Feito isso, toda a codificação da tela é feita no arquivo .kv como pode ser visto abaixo.

A comunicação entre o arquivo Python e o arquivo Kivy se dá através de que no Kivy na sua inicialização procura um arquivo chamado com o mesmo nome da classe App no Python, que no nosso caso se chama “Bibliotec”.

```
<Emprestimo1>
```

```
FloatLayout:
```

```
Image:
```

```
    source: 'livros.jpg'
    size_hint_y: None
    height: 700
    allow_stretch: True
    pos_hint: {'x': .25}
```

```
Label:
```

```
    text: 'Empréstimos'
    font_size: 50
    pos_hint: {'x': -.2, 'y': .35}
```

```
Button:
```

```
    background_normal: 'white'
    color: 'black'
    text: 'Devolver Livro'
    on_release: app.root.current = 'emprestar'
    on_press: app.delete()
    pos_hint: {'x': .15, 'y': .5}
    size_hint: (0.3, 0.1)
    font_size: 35
```

```
Button:
```

```
    background_normal: 'white'
    color: 'black'
    text: 'Realizar Empréstimo'
    on_release: app.root.current = 'emprestar'
    on_press: app.insert()
    pos_hint: {'x': .15, 'y': .65}
    size_hint: (0.3, 0.1)
    font_size: 35
```

```
Button:
```



```

background_normal: 'white'
color: 'black'
text: 'Voltar ao menu'
on_release: app.root.current = 'menu'
on_press: app.clear_text()
size_hint: (0.1, 0.1)
pos_hint: {'x': .05, 'y': .05}

```

O tipo de layout utilizado para as telas foi o FloatLayout que permite um maior aproveitamento da tela, podendo ser determinada a posição de cada um dos elementos da tela através do comando “pos_hint”.

Por último, foi feita a integração da interface com o banco de dados. Para isso, foi utilizado a biblioteca Python psycopg2, que possui funções perfeitas para nosso objetivo, como, por exemplo, a cursor.execute, que executa uma linha de código SQL e a fetchall que armazena o que foi retornado nessa linha, no código apresentado abaixo é a função para conexão com o pgAdmin4.

```

connection = psycopg2.connect(user="postgres",
                                password="ms1990id41",
                                host="127.0.0.1",
                                port="5432",
                                database="postgres")

```

3. Resultados

Nas figuras 2 a 9 abaixo serão mostradas as telas codificadas, vale ressaltar que houveram algumas variações do protótipo que desenvolvemos.



Figura 2 - Tela Inicial da Interface
Fonte: Os Autores.

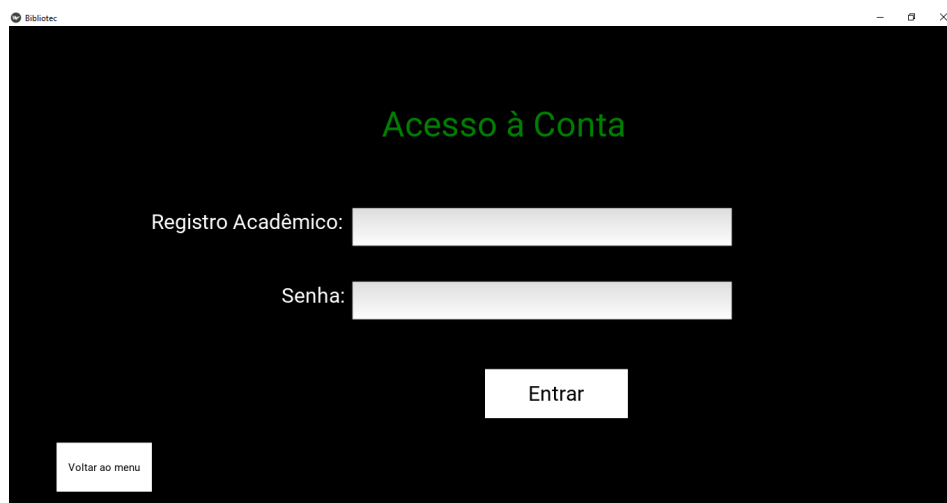
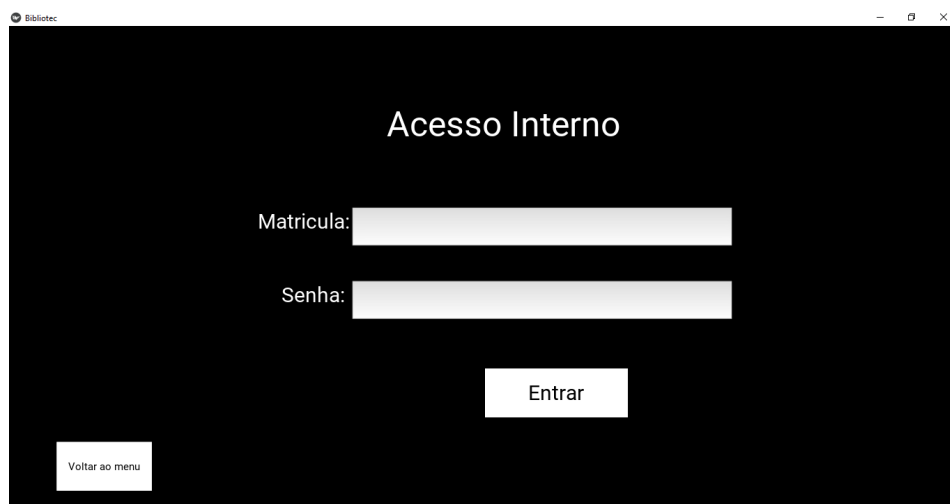


Figura 3 - Tela de acesso aos estudantes
Fonte: Os Autores.



Figura 4 - Menu para estudantes
Fonte: Os Autores.



The screenshot shows a web browser window with the title 'Biblotec'. The main heading is 'Acesso Interno'. Below it, there are two input fields: 'Matricula:' and 'Senha:'. A white 'Entrar' button is positioned below the password field. In the bottom left corner, there is a 'Voltar ao menu' button.

Figura 5 - Tela de acesso para funcionários
Fonte: Os Autores.



The screenshot shows a web browser window with the title 'Biblotec'. The main heading is 'Acesso Interno'. On the left side, there are three buttons stacked vertically: 'Cadastrar Aluno', 'Cadastrar Livro', and 'Realizar Consulta'. A 'Voltar ao menu' button is located in the bottom left corner. On the right side of the screen, there is a photograph of a stack of books with one book open on top.

Figura 6 - Menu para funcionários
Fonte: Os Autores.

Cadastro de Estudantes

Nome: RA:

E-mail:

Curso:

Senha:

Figura 7 - Tela de cadastro de estudante
Fonte: Os Autores

Cadastro de Livros

Título: Código:

Autor:

Categoria:

Figura 8 - Tela de cadastro de livros
Fonte: Os Autores.

ID	Título	RA	Nome	Horário de empréstimo	Data de devolução
91180	Stan Helsing	1572753	Gorden Moulit	2021-12-19 16:01:47.949847	2021-12-26
79000	Captain Newman, M.D.	1978896	Luiza Stringhini	2021-12-19 15:24:12.017582	2021-12-26
93175	Ikigami	1723972	Inesita Sewall	2021-12-19 12:34:55.949769	2021-12-26
91306	The Devil Thumbs a Ride	2111515	Diego Cheorhe	2021-12-19 10:11:45.299164	2021-12-26
90557	Stolen (Stolen Lives)	1611036	Arda Brownbridge	2021-12-19 07:46:19.458746	2021-12-26
88142	Skeletons	1915545	Boyd Rignoldes	2021-12-19 02:09:44.293994	2021-12-26
80925	Trip to Bountiful, The	1907822	Rafaelita Buntine	2021-12-18 09:47:53.588239	2021-12-25
95408	Freedom	1631180	Lonnie Fries	2021-12-17 23:42:13.918191	2021-12-24
83368	Dupes, The (Al-makhdu'un)	2104969	Antonin Pardal	2021-12-17 06:51:24.233158	2021-12-24
80617	Garden of Eden, The	1965500	Cecelia Bosma	2021-12-16 20:02:47.004052	2021-12-23
82045	American Buffalo	2152398	Sibby Vennings	2021-12-16 15:40:34.063818	2021-12-23
85769	Bastards of the Party	1715034	Editha Meijer	2021-12-16 15:38:53.704901	2021-12-23
85916	Michael Jackson: Life of a Superst	1902680	Ashlen Yandell	2021-12-16 14:57:22.665819	2021-12-23
81800	Big Easy Express	2026254	Britta Summerton	2021-12-15 21:43:18.685915	2021-12-22
84621	Lola	1592714	Charmine Caurah	2021-12-15 20:09:22.278514	2021-12-22
93665	Santa Claus Conquers the Martians	1726636	Sarina Hodgins	2021-12-15 19:47:51.614074	2021-12-22
88891	Forsaken, The	2123241	Antone Eriksson	2021-12-15 16:36:58.471752	2021-12-22
88366	Prata Palomares	1934632	Muire Noel	2021-12-15 16:28:48.258935	2021-12-22
92580	Time Freak	2122024	Neel Brecon	2021-12-15 05:40:07.588762	2021-12-22
93994	Miss Congeniality 2: Armed and Fabul	1720001	Walker Wyatt	2021-12-14 19:41:29.914556	2021-12-21
88765	Bonnie and Clyde	2046615	Annnora Kingham	2021-12-14 19:34:43.435892	2021-12-21

Figura 9 - Tela de consulta de empréstimos
Fonte: Os Autores.

Na interface, também foram codificadas Pop-ups , para isso foi feito o tratamento de erros, para informar ao usuário o erro que ele cometeu, ou também para informar que as operações foram concluídas com sucesso, em seguida serão apresentadas as Pop-ups.

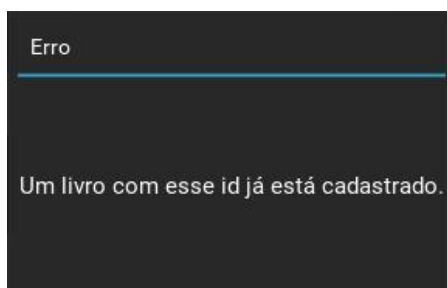


Figura 10 - Pop-up quando é cadastrado um livro que já existe no banco
Fonte: Os Autores

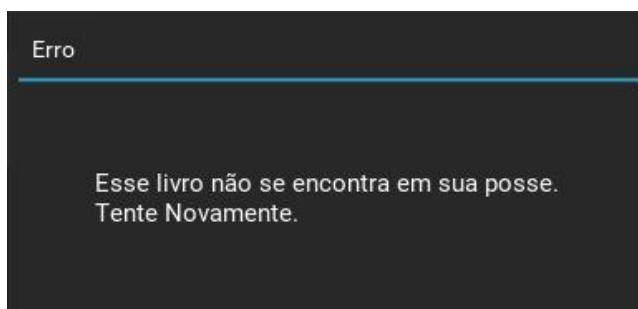


Figura 12 - Pop-up de aviso quando há a tentativa de devolver um livro que não está com o respectivo RA
Fonte: Os Autores

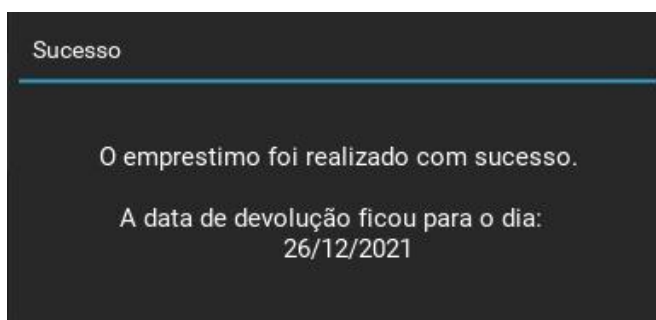


Figura 13 - Pop-up quando o empréstimo é realizado com sucesso
Fonte: Os Autores

4. Conclusão

Neste projeto foi possível aprender sobre outras alternativas de utilizar a linguagem Python, que é no desenvolvimento de interfaces gráficas, mais precisamente sobre o Python Kivy, juntamente com isso aprendemos a aplicar outras funções dele, também, como a integração com um banco de dados e a utilizar a biblioteca pyzbar para escanear os códigos de barras dos livros. No desenvolvimento da aplicação também foi possível colocar em prática os conhecimentos adquiridos nas disciplinas de Banco de Dados 1 e 2 para o desenvolvimento do Backend.

5. Referências

- [1] PostgreSQL - [PostgreSQL 14.1 Documentation](#) - Acesso em 26 de outubro de 2021
- [2] Documentação Pyzbar - [pyzbar · PyPI](#) - Acesso em 6 de novembro de 2021
- [3] Playlist HashLDash - Python Kivy [1 - Python Kivy - criando uma interface gráfica](#) - Acesso em 11 de novembro de 2021
- [4] Documentação Kivy - [Welcome to Kivy — Kivy 2.0.0 documentation](#) - Acesso em 11 de novembro de 2021
- [5] Documentação Psycopg2 - [Psycopg – PostgreSQL database adapter for Python — Psycopg 2.9.3.dev0 documentation](#) - Acesso em 25 de novembro de 2021