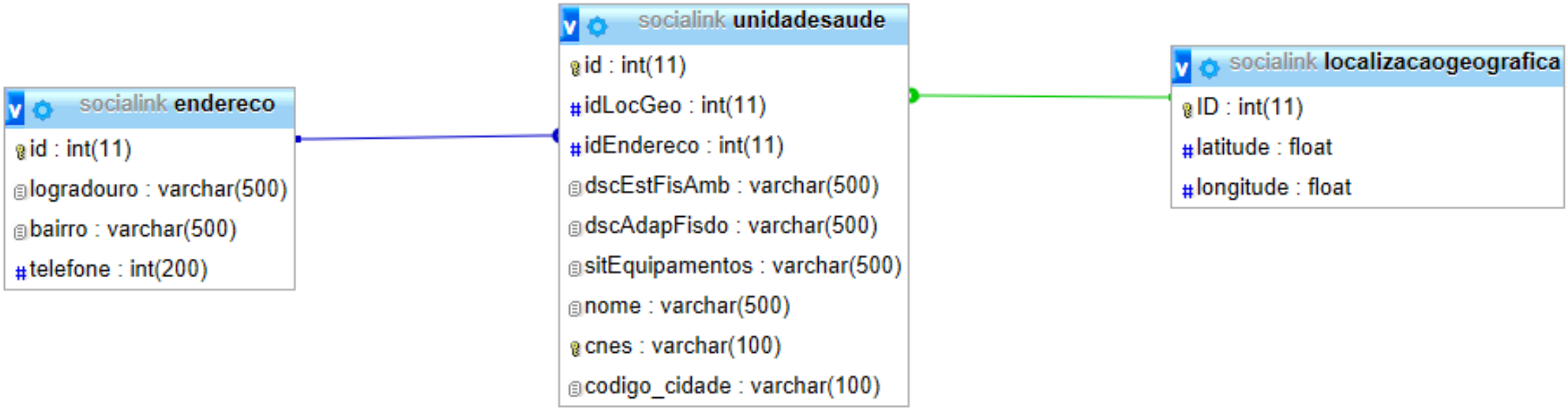
GUI com Tkinter e Banco de Dados com MySQL

Python

Modelo de Dados



Conexão com o Banco de Dados

- host: localhost
- user: root
- Password: ?
- Database: socialink

Conexão com o Banco de Dados

- Padrão Python para interfaces de bancos de dados: Python DB-Py
- A maioria das interfaces para banco de dados adere a este padrão
  - MySQL
  - PostgreSQL
  - Microsoft SQL Server
  - Oracle
  - SQLite

Conexão com o Banco de Dados

- Instalando driver necessário
  - pip install PyMySQL
- Abrindo conexão
  - db = pymysql.connect("localhost", "root", "", "socialink")
  - cur = db.cursor()
  - con.execute('insert into LocalizacaoGeografica(latitude, longitude) values(%s, %s)' % (-10.911237001414188, -37.062077522276 8))
  - db.commit()

Preparando Transações

- As transações são um mecanismo que garante a consistência dos dados. As transações têm as quatro propriedades seguintes:
- Atomicity: Ou uma transação é concluída ou nada acontece.
- Consistency: A transação deve começar em um estado consistente e deixar o sistema em um estado consistente.
- Isolation: resultados intermediários de uma transação não são visíveis fora da transação atual.
- Durability: Uma vez que uma transação foi confirmada, os efeitos são persistentes, mesmo após uma falha do sistema.
- O Python DB API 2.0 fornece dois métodos para ambos os commit ou rollback uma transação.

Preparando Transações

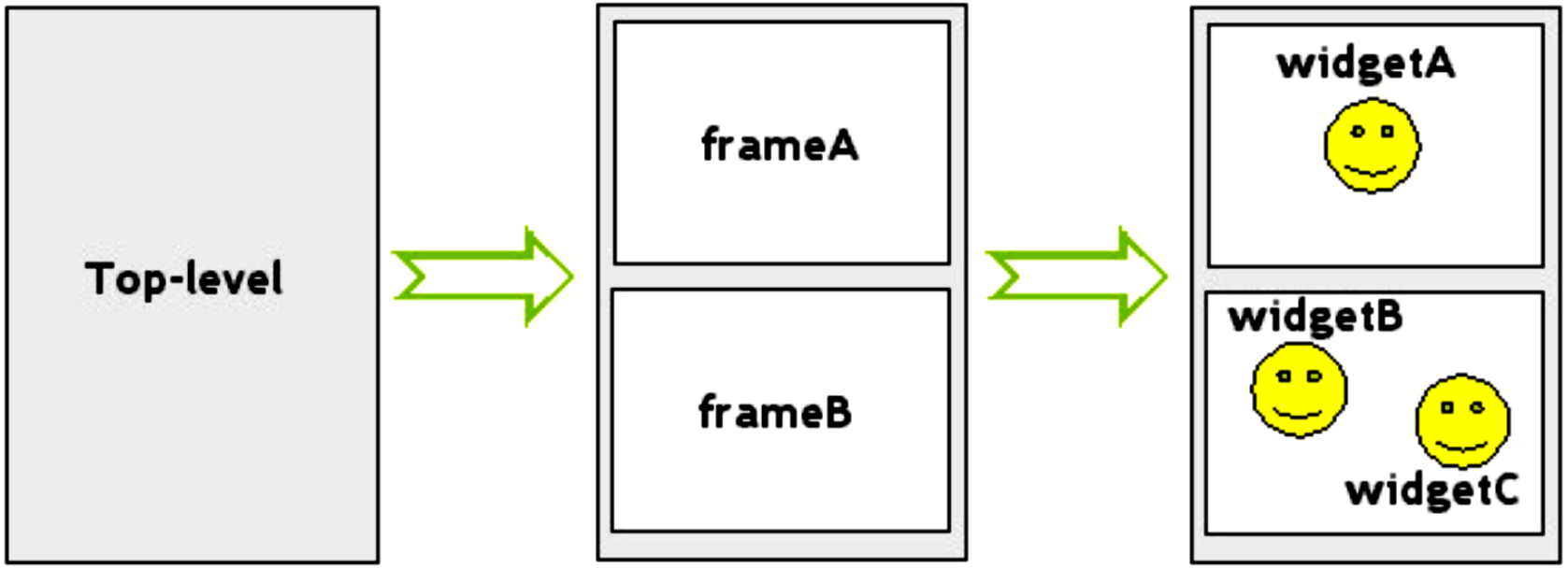
```
# Prepare SQL query to DELETE required records
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # Execute the SQL command
   cursor.execute(sql)
   # Commit your changes in the database
   db.commit()
except:
   # Rollback in case there is any error
   db.rollback()
```

Tkinter em Python

- Widgents
- Event Handlers
- bidings

Tkinter em Python

- Widgents
- Event Handlers
- bidings



CÓDIGO:

```
frameA(Top-level)
frameB(Top-level)
widgetA(frameA)
widgetB(frameB)
widgetC(frameB)
```

Conclusão

- Apresentar código do repositório