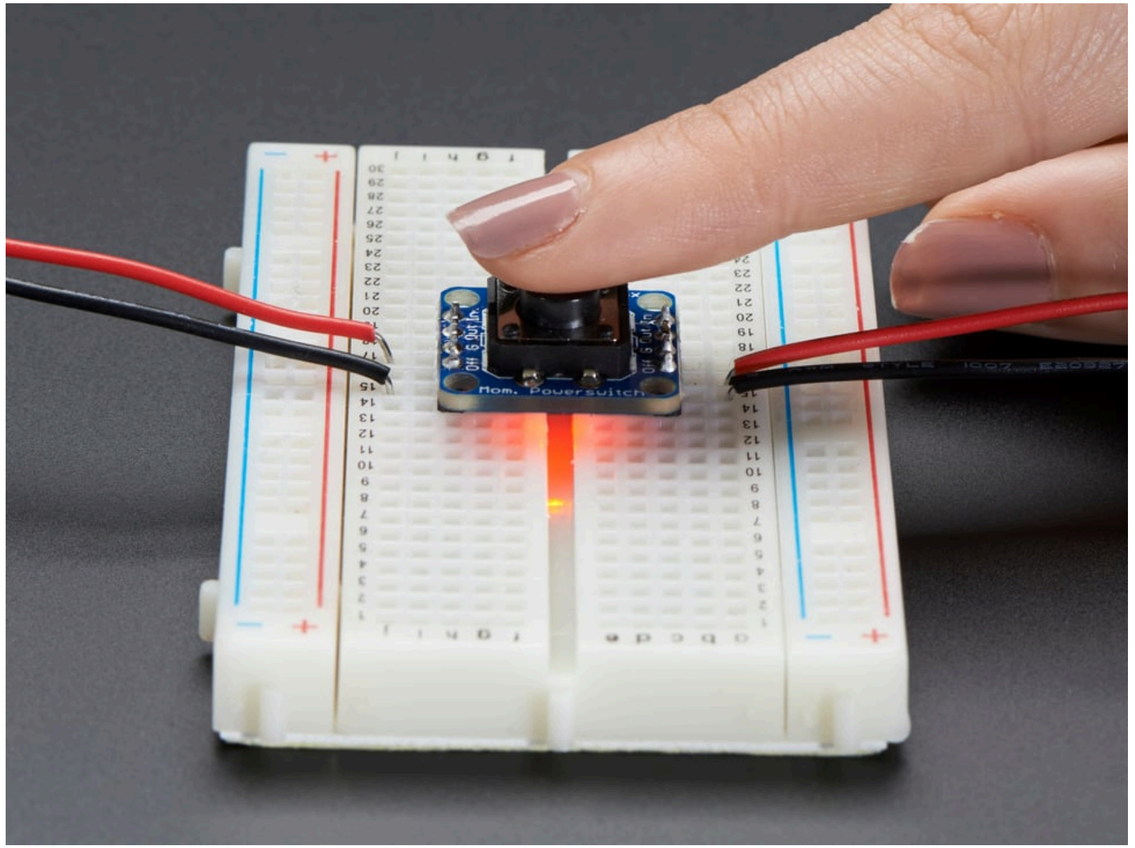
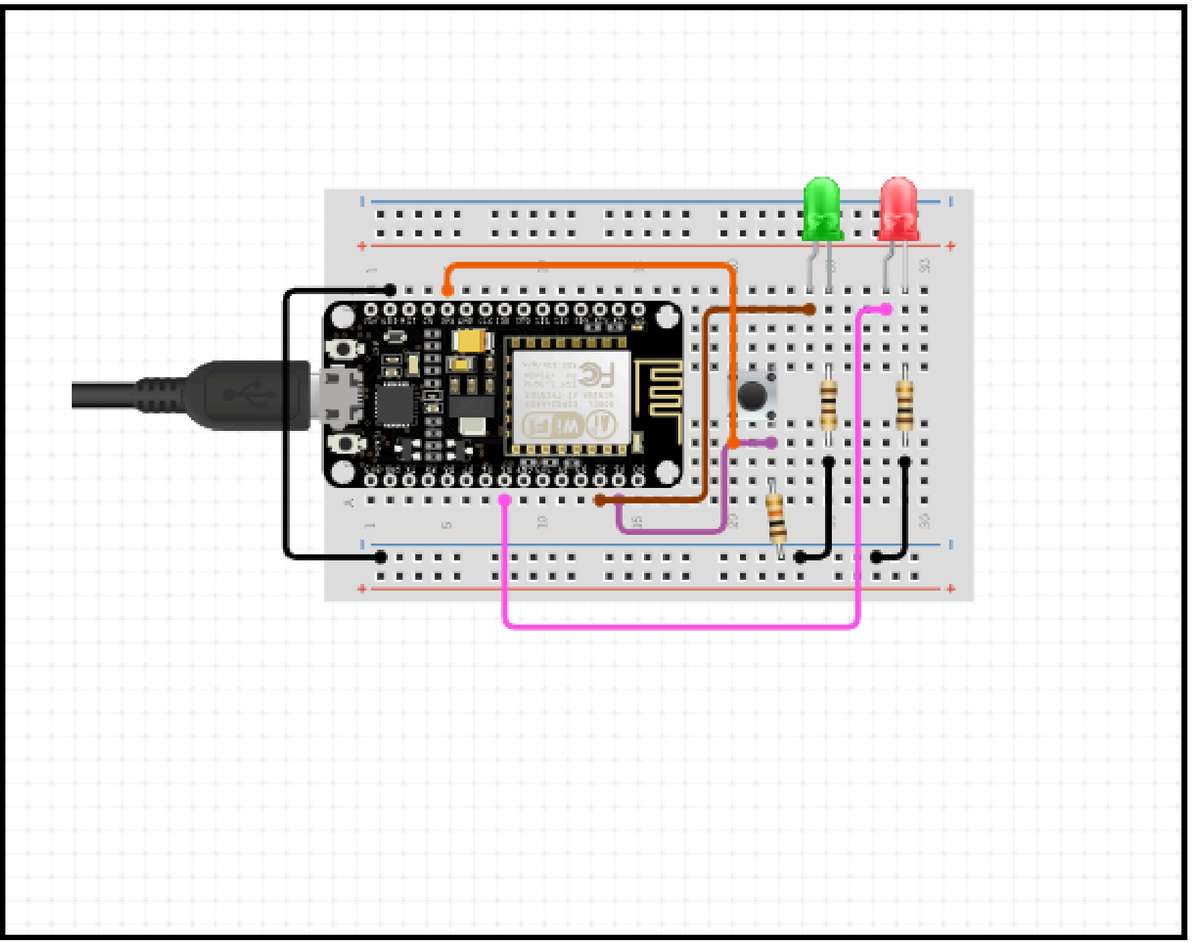
Reaction - Projeto Embarcados

Apresentação por Bruno Q. e Felipe

Resumo do Projeto

Criar um sistema cujo objetivo é medir o tempo de reação do usuário, registrar o histórico de reações e possibilitar a configuração do teste.





Hardware

- ESP8266
- LED Basic Green 5mm
- LED Basic Red 5mm
- Mini Pushbutton Switch

```
const char* ssid = "wifi";
const char* password = "senha";
const char* serverUrl = "https://aps-reaction.onrender.com";
WiFiClientSecure client;

bool jogoAtivo = false;
unsigned long inicioTempo = 0;
int minDelay = 500;
int maxDelay = 5000;

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  Serial.print("Conectando ao WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConectado ao WiFi!");

  client.setInsecure();

  pushButton.init();

  Serial.println("=== Jogo de Reação ===");
  Serial.println("Digite '1' para jogar ou '2' para ver o Top 10.");
}
```

Setup

- ssid e password
- API no Render
- Loop até conectar
- Declaração da config padrão

```
void loop() {
  if (Serial.available() > 0) {
    char comando = Serial.read();
    if (comando == '1' && !jogoAtivo) iniciarJogo();
    else if (comando == '2') listarTop10();
    else if (comando == '3') obterConfig();
  }

  if (jogoAtivo && pushButton.read() == 0) {
    delay(50);
    if (pushButton.read() == 0) {
      unsigned long tempoReacao = millis() - inicioTempo;
      ledG.off();
      enviarReacao(tempoReacao);
      jogoAtivo = false;
      Serial.println("Digite '1' para jogar novamente.");
    }
  }
}
```

Loop

- Iniciar jogo
- Listar Top10
- obter Config
- Jogar novamente

```
void iniciarJogo() {
  obterConfig();
  jogoAtivo = true;

  Serial.println("Prepare-se...");
  for (int i = 3; i > 0; i--) {
    Serial.println(i);
    ledR.on(); delay(300);
    ledR.off(); delay(300);
  }

  delay(random(minDelay, maxDelay));
  Serial.println("Aperte!!");
  ledG.on();
  inicioTempo = millis();
}
```

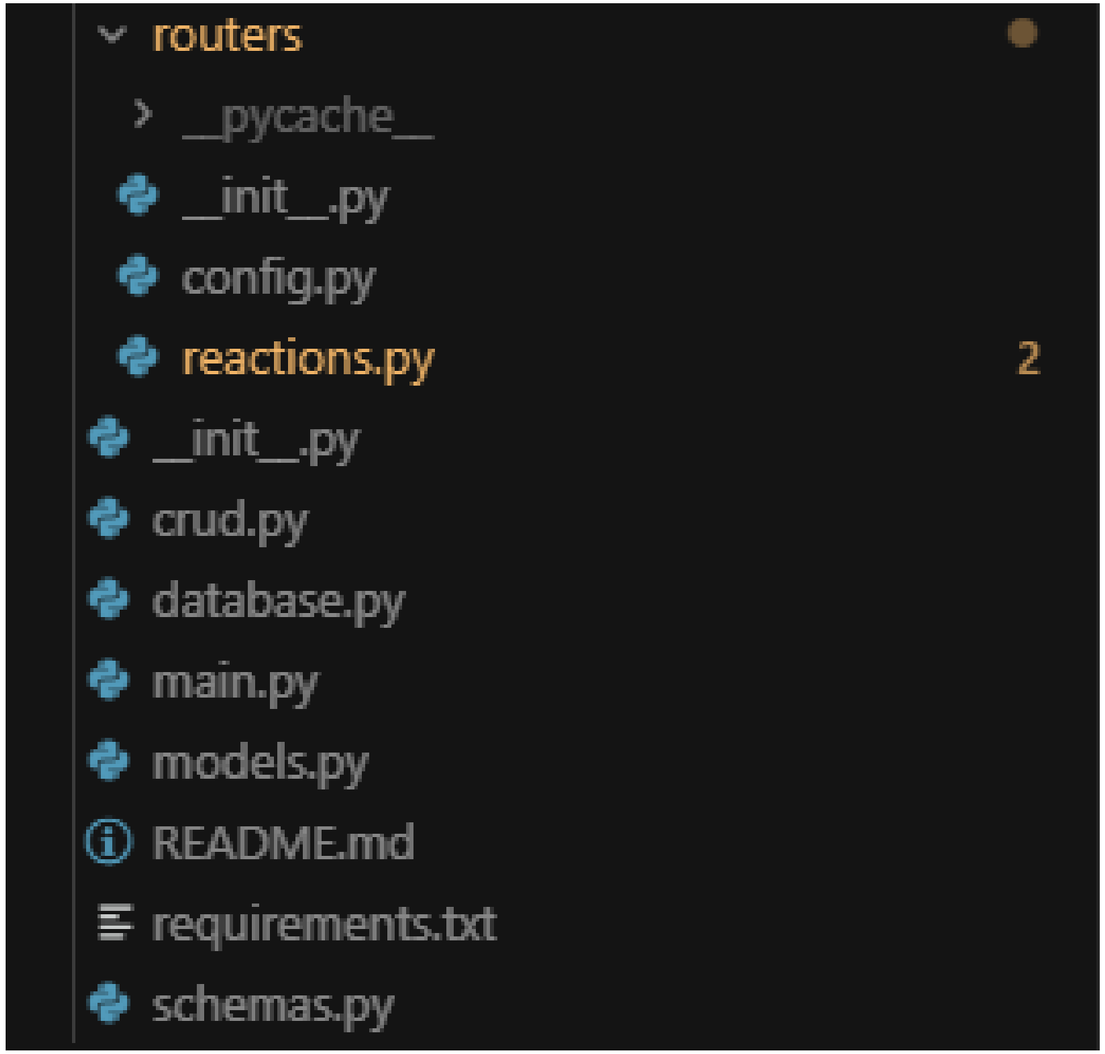
Iniciar Jogo

- Obtém config
- Led RED indica início
- Randomiza o delay
- Millis()

```
void enviarReacao(unsigned long tempo) {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(client, String(serverUrl) + "/reactions/");
    http.addHeader("Content-Type", "application/json");

    String body = "{\"reaction_time\": " + String(tempo) + "}";
    int code = http.POST(body);

    Serial.println("Código de retorno do envio de tempo:" + String

    if (code == 200) {
      Serial.println("Tempo enviado com sucesso!");
    } else {
      Serial.println("Falha ao enviar tempo.");
    }

    http.end();
    client.stop();
  }
}
```

HTTP

- Veirifica se está conectado
- Pega URL do render
- Cria Header e Body
- Chama http.post() ou http.get()



API

- FastAPI
- Routes
- CRUD(repository)
- Models e Schemas

```python
@router.post("/", response_model=schemas.ReactionOut)
def create_reaction(reaction: schemas.ReactionCreate, db: Session = D
    return crud.create_reaction(db, reaction)


@router.get("/top10", response_model=list[schemas.ReactionOut])
def get_top10(db: Session = Depends(get_db)):
    return crud.get_top_reactions(db)

@router.get("/last", response_model=schemas.ReactionOut)
def get_last_reaction(db: Session = Depends(get_db)):
    return crud.get_last_reaction(db)

@router.get("/last/{count}", response_model=list[schemas.ReactionOut]
def get_last_reactions(count: int, db: Session = Depends(get_db)):
    return crud.get_last_reactions(db, count)

@router.get("/{id}", response_model=schemas.ReactionOut)
def get_reaction(id: int, db: Session = Depends(get_db)):
    return crud.get_reaction_by_id(db, id)
```

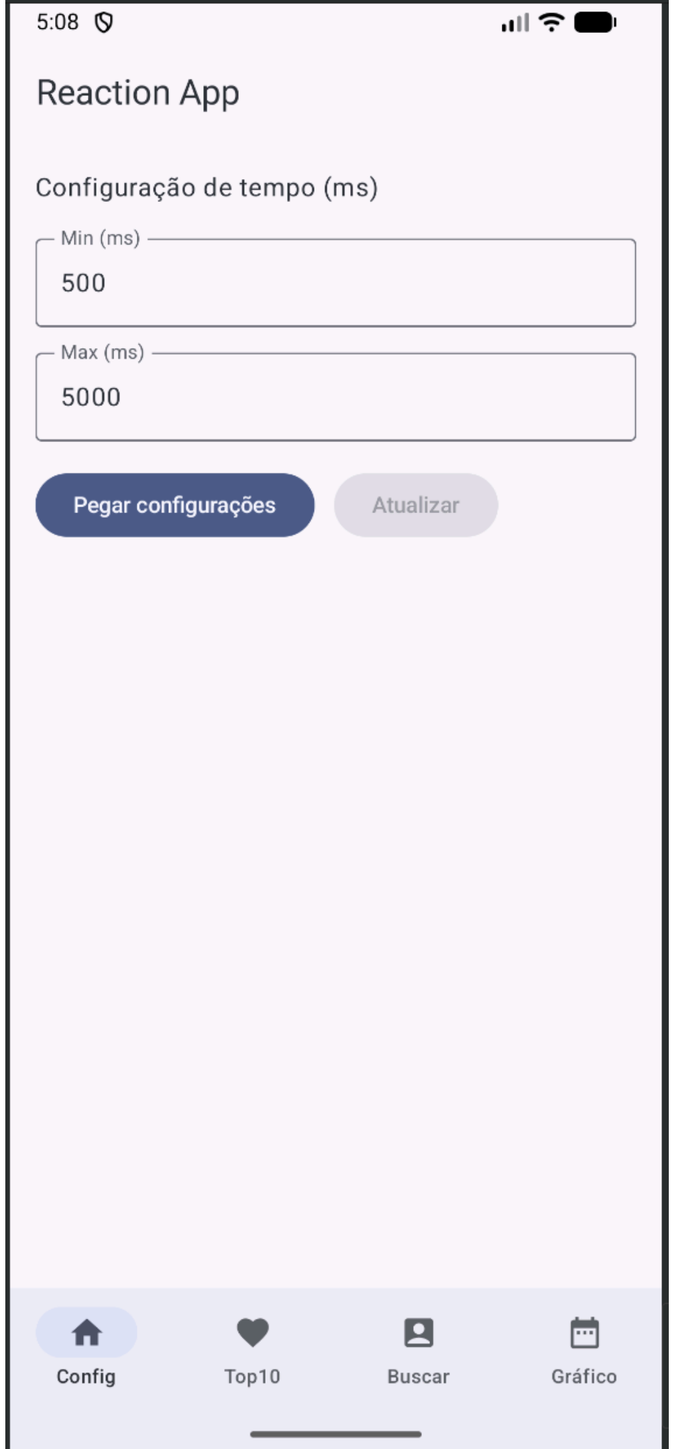
Routes

- Post para create reaction
- top10
- last e last{count}
- {id}

```python
def create_reaction(db: Session, reaction: schemas.ReactionCreate):
    db_reaction = models.Reaction(reaction_time=reaction.reaction_ti
    db.add(db_reaction)
    db.commit()
    db.refresh(db_reaction)
    return db_reaction

def get_top_reactions(db: Session, limit: int = 10):
    return db.query(models.Reaction).order_by(models.Reaction.reacti

def get_last_reaction(db: Session):
    return db.query(models.Reaction).order_by(models.Reaction.timest

def get_last_reactions(db: Session, count: int):
    return (
        db.query(models.Reaction)
        .order_by(models.Reaction.timestamp.desc())
        .limit(count)
        .all()
    )
```

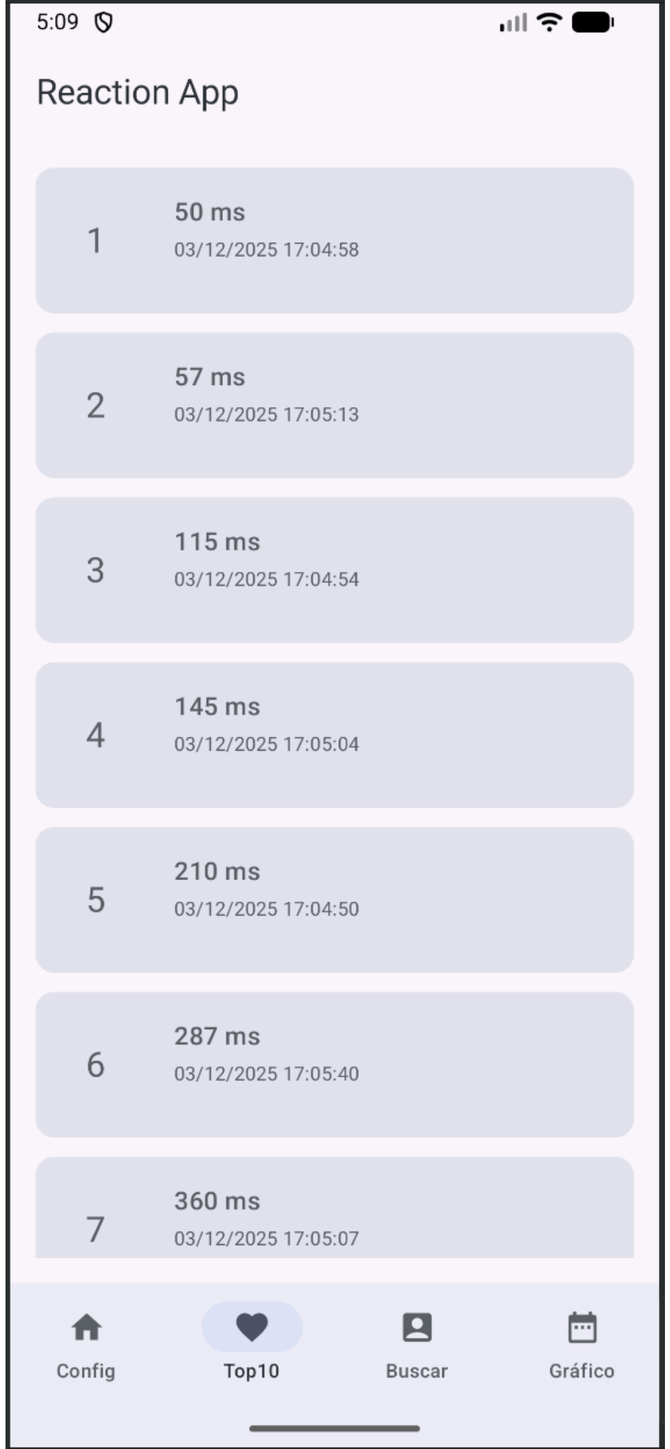
CRUD

- Usa o session do Routes
- cria um objetivo Reaction
- commit()
- query()



APP

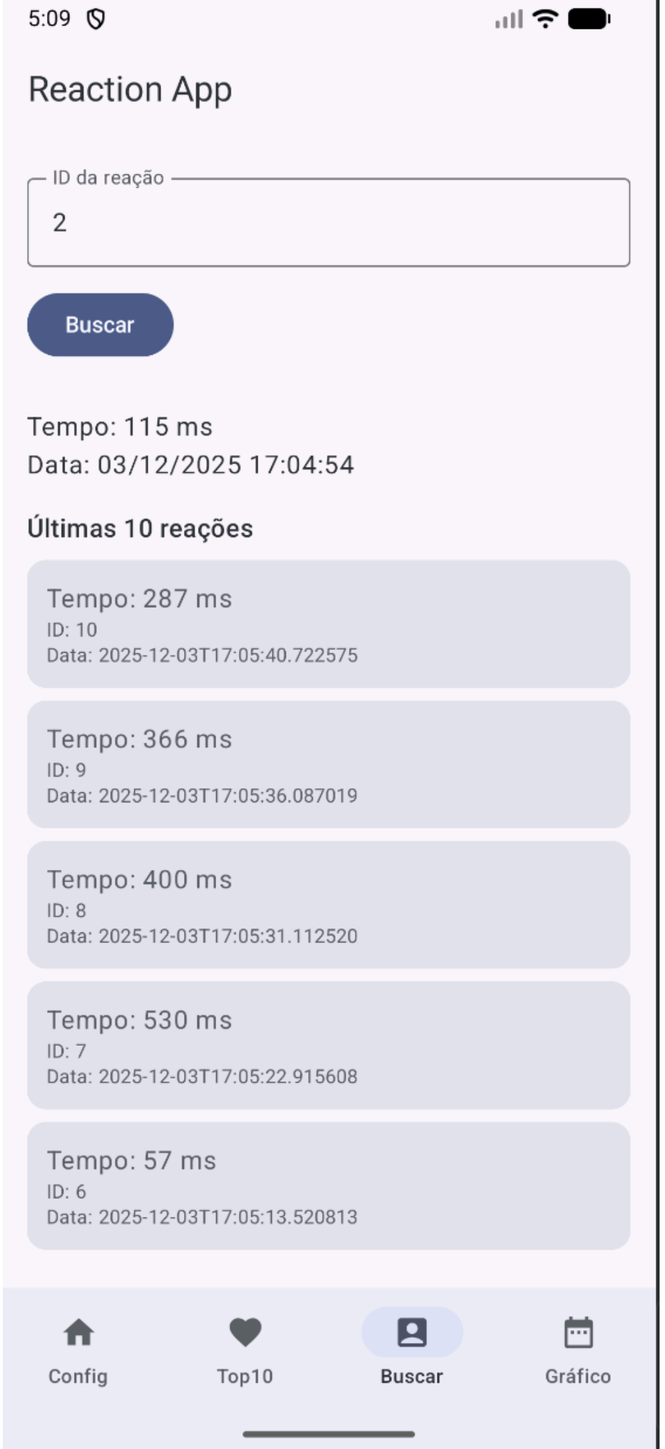
- Configuração de tempo
- Top 10 reações
- Buscar
- Gráfico



Top 10

Rank ordenado por tempo

Data de quando foi feita



Buscar Reação

Busca feita por ID

Gráfico de reações

Últimas 10 reações feitas