Sistemas Distribuídos
Trabalho Final

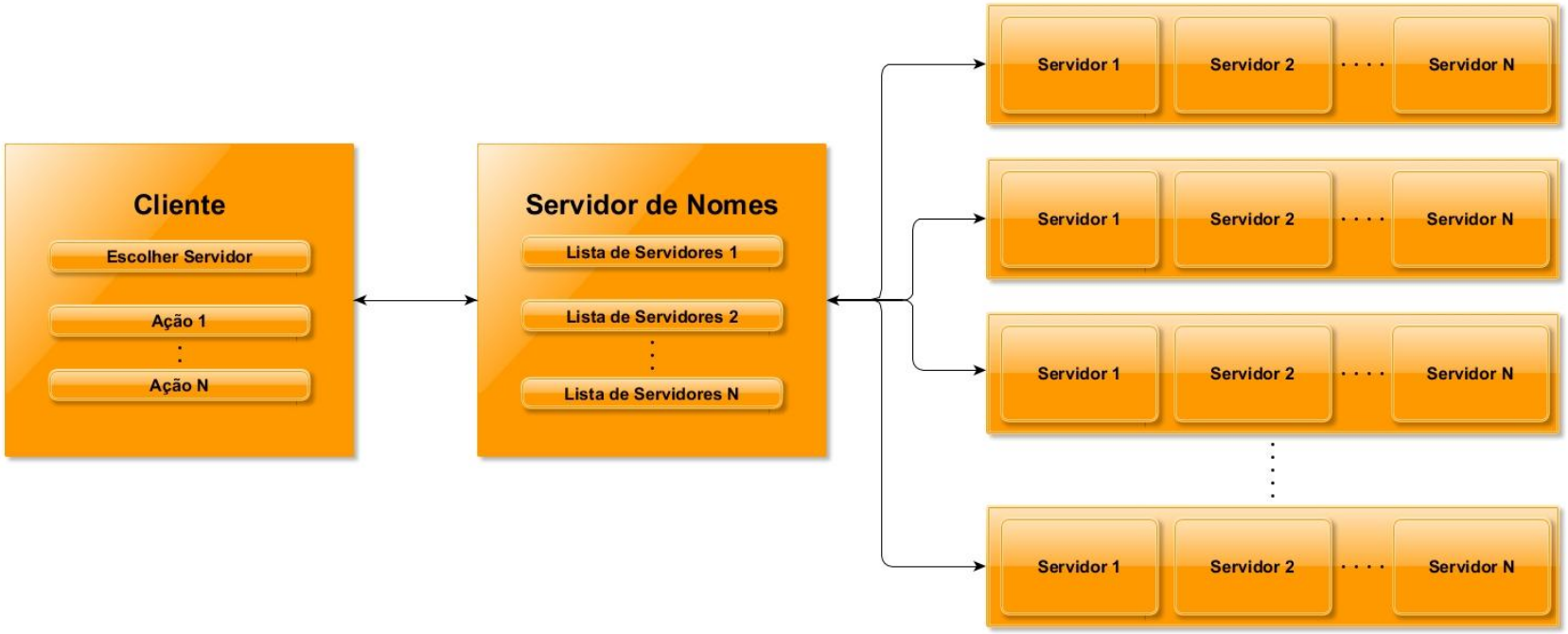
Servidor de Serviços Distribuídos

Alunos:
Vinícius Gabriel Santos
Luiz Gustavo Mattos

Motivação e Especificações

- Motivação para o Desenvolvimento do Trabalho
- Principais Preocupações
- Métodos Utilizados
- Linguagens Utilizadas - Python, SQL e Ruby
- Serviços são Facilmente Acoplados
- Arquitetura Centralizada

Arquitetura do Servidor

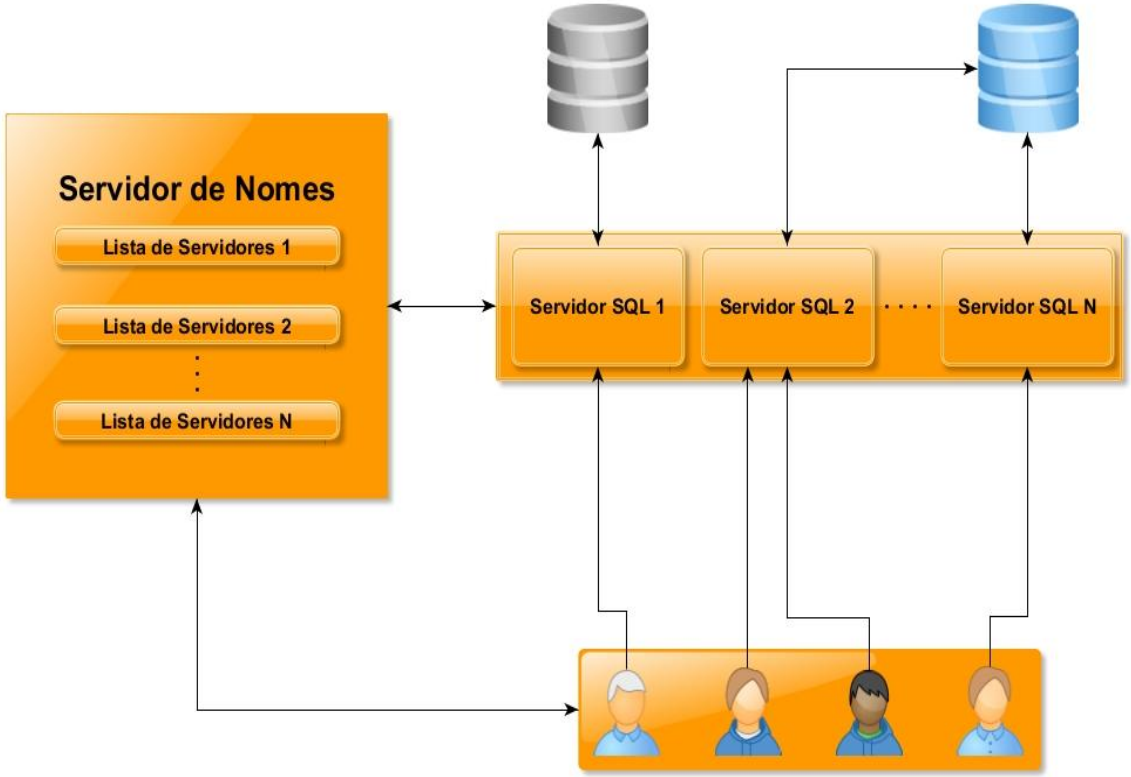


Serviços Implementados

- Servidor de Nomes
- Threadpool
- Servidor SQL
- Servidor Publish-Subscribe

Funcionamento

- Interação Servidor SQL, comunicação com Banco de Dados.
- Comunicação de usuários e ações dos usuários SQL.



Principais Funções do Servidor de Serviços

Métodos Principais - Cliente

```python
def run(self):
        op = 1

        self.createSocketUDP()

        while op != 0:
            op = self.getService()

            print("Serviços disponíveis")

            for i in range(len(self.serviceList)):
                print(i+1, "- ", self.serviceList[i])

            print("0 - Sair")

            op = int(input("Selecione o serviço: "))

            while op < 0 or op > len(self.serviceList):
                op = int(input("Serviço invalido, digite novamente: "))

            self.selectService(op)
```

Métodos Principais - Cliente

```
def getService(self):

        self.sendUDP((DNS_IP, DNSPORT), self.prepareMsg("getServices"))
        data,  = self.sock.recvfrom(1024)

        self.serviceList = self.loadMessage(data)
```

Métodos Principais - Servidor de Nomes

```
def getAddress(self, data, address):
        message = self.loadMessage(data)
        hasService = False

        if message == "getServices":
            self.sendToHost(address, self.services)
        else:
            for i in range(len(self.services)):
                if self.services[i] == message["type"]:
                    hasService = True
                    if message["con"] == "SERVER":
                        print("Adding ", message["type"], " server.")
                        self.addQueueSv(i, address)
                        print("DONE!")
                        self.sendToHost(address, "DONE!")
                    else:
                        svAddr = self.getServerAddress(i)

                        self.sendToHost(address, svAddr)

            if not hasService:
                self.sendToHost(address, "ERROR!")

        self.threads.repopulate()
```

Métodos Principais

Servidor SQL

```
def run(self, connection):
        sqlConnector = mysql.connector.connect(
            host="localhost",
            user=dbUser,
            passwd=dbPass,
            database="sqlDB"
        )

        cursor = sqlConnector.cursor()

        msg = self.getMessage(connection)

        if msg == "login":
            connection.send(self.prepareMsg("Digite seu login: "))
            login = self.getMessage(connection)
            connection.send(self.prepareMsg("Digite sua senha: "))
            passwd = self.getMessage(connection)

            sql = "SELECT * FROM funcionarios where login = %s and senha = %s"
            val = (login, passwd)

            cursor.execute(sql, val)

            results = cursor.fetchall()

            if len(results) == 1:
                self.menu(connection, cursor, sqlConnector)
            else:
                connection.send(self.prepareMsg("Falha na autenticação"))

        sqlConnector.close()

        self.threads.repopulate()
```

Métodos Principais - Chat

Publish-Subscribe

```
Thread.new{
  begin
    queue.subscribe(block: true) do |_delivery_info, _properties, body|
      puts body
      comLogs << body << "\n"
    end
  rescue Interrupt => _
    channel.close
    connection.close
  end
}

message = argvMessage.empty? ? (argvMessage = gets) : argvMessage

newMessage = "[#{name}] #{message}"

exchange.publish(newMessage)

while !(message.include? "exit")

  message = gets.chomp

  newMessage = "[#{name}] #{message}"

  exchange.publish(newMessage)


end
```

Contextualização

A arquitetura apresentada pode ser facilmente adaptada para um ambiente empresarial, facilitando os acessos e controlando-os.

Dúvidas? Perguntas?

Obrigado!