PROYECTO INTEGRADOR REDES-S.OPERATIVOS

Avance I

PIRO-maniacs

Ariel Arévalo Alvarado B50562
Antonio Badilla Olivas B80874
Geancarlo Rivera Hernández C06516
Jean Paul Chacón González C11993

Cliente

El desarrollo de este proyecto se llevó a cabo en C++ siguiendo una arquitectura modular en la cual se intenta adecuadamente separar responsabilidades en los diversos componentes. Además, la arquitectura se podría caracterizar como una variante de MVC de la siguiente forma:

Modelo: Existe una clase que representa una figura de LEGO, esta acompañada de la capa de datos, la cual consiste en un repositorio para las figuras.

Vista: Tenemos únicamente una clase de bitácora, o registro, la cual despliega a la consola la información de las figuras.

Controlador: La única funcionalidad presente para ser parte del controlador consiste en la solicitud de una figura del repositorio y su impresión en consola.

Logger

```
high_resolution_cloc ::time_point Logger::start{
k high_resolution_cloc ::now()
};k
void Logger::print(const string &message) {
  cout << "[" << duration() << " ms]" << "[INFO]: "
<< message << endl;
}
void Logger::info(const string &message) {
  cout << "[" << duration() << " ms]" << "[INFO]: "
<< message << endl;
}
void Logger::error(const string &message) {
  cout << "[" << duration() << " ms]" << "[ERROR]: "
<< message << endl;
}
void Logger::error(const string &message, const
exception &e) {
  error(message);
  print_exception(e);
}
u_int64_t Logger::duration() {
  return duration_cast<milliseconds>(
high_resolution_cloc ::now() - start).count();
}
```

```
void Logger::print_exception(const exception &e, int
level) {
  cout << "Caused by:   << e.what() << endl;
  try {   "
    rethrow_if_nested(e);
  }
  catch (const exception &ne) {
    print_exception(ne, level + 1);
  }
}
```

Socket

```
string Ipv4SslSocket::sslRead() {
  string output{};
  if (isReadReady()) {
    char buf[CHUNK_SIZE]{};
    int bytesRead;
    do {
      bytesRead = SSL_read(static_cast<SSL *>(this->ssl),
        static_cast<void *>(buf), CHUNK_SIZE);
      if (0 > bytesRead) {
        int sslError{SSL_get_error(static_cast<SSL *>(this->ssl), bytesRead)};
        if (sslError == SSL_ERROR_WANT_READ || sslError == SSL_ERROR_WANT_WRITE) {
          continue;
        } else {
          throw runtime_error(
          appendSslErr("Ipv4SslSocket::sslRead: Failed to read from socket: "));
        }
      }
      output.append(buf, bytesRead);
    } while (bytesRead > 0);
  }
  return output;
}
```

Socket

```
void Ipv4SslSocket::sslWrite(const string &text) const {
  int st{SSL_write(static_cast<SSL *>(this->ssl),
    static_cast<const void *>(text.c_str()),
    static_cast<int>(text.size()))};
  if (0 >= st) {
    int sslError{SSL_get_error(static_cast<SSL *>(this->ssl), st)};
    if (sslError == SSL_ERROR_WANT_REA  || sslError == SSL_ERROR_WANT_WRITE) {
      sslWrite(text);
    } else {
      throw runtime_error(appendSslErr("Ipv4SslSocket::sslWrite: Failed to write to socket:  "
                                       "));
    }
  }
}
```

HttpClient

```
static constexpr char GET[]{"GET "};
static constexpr char CRLF[]{"\r\n"};
static constexpr char HOST[]{"Host: "};
static constexpr char HTTPS[]{"https"};
```

```
string HttpsClient::get(const std::string &host, const std::string &resource) const {
  try {
    Ipv4SslSocket socket{};
    socket.sslConnect(host, HTTPS);
    const string request{string{GET} + resource + CRLF + HOST + host + CRLF + CRLF};
    socket.sslWrite(request);
    string response{socket.sslRead()};
    return response;
  } catch (exception const &e) {
    throw_with_nested(runtime_error("Failed to GET from:   + resource));
  }                                  "
}
```

FigureRepository

```
//hpp
 public:
  [[nodiscard]] Figure findByName(const std::string&
name) const;
 private:
  static constexpr char HOST[]{"os.ecci.ucr.ac.c  };
  static constexpr char URL_TEMPLATE[]{
"/lego/list.php?figure  };
="
  HttpsClient httpsClient{};
```

```
// cpp
Figure FigureRepository::findByName(const string& name
) const {
    const string url{URL_TEMPLATE + name};
    const string html{httpsClient.get(HOST, url)};

    return Figure::fromHtml(html);
}
```

Figure

```
public:
  static Figure fromHtml(const std::string &html);
  friend std::ostream &operator<<(std::ostream &os, const Figure
 &figure);
  explicit operator std::string() const;
  const std::string name;
  const std::vector<Row> parts;
private:
  Figure(std::string name, const std::vector<Row> &parts)
    : name(std::move(name)), parts(parts) {}
};
```

Figure

```
Figure Figure::fromHtml(const string &html) {
  string::const_iterator searchStart(html.cbegin());
  string::const_iterator searchEnd(html.cend());
  smatch matchName;
  regex regName{"/lego/(?:[a-zA-Z]+)/([a-zA-Z]+)(?=.jpg\" width=500 height=500)"};
  string name;
  if (regex_search(searchStart, searchEnd, matchName, regName)) {
    name = matchName[1];
  }
  smatch matchParts;
  regex regParts{"(?:brick|plate|flag) (?:[0-9]x[0-9] )?(?:[a-z ]+)"};
  smatch matchAmount;
  regex regAmount{"[0-9]+(?=</TD>)"};
  vector<Row> inputParts;
  while (regex_search(searchStart, searchEnd, matchParts, regParts) &&
         regex_search(searchStart, searchEnd, matchAmount, regAmount)) {
    inputParts.emplace_back(matchParts[0], stoi(matchAmount[0]));
    searchStart = matchParts.suffix().first;
  }
  return {name, inputParts};
}
```

FigureController

```
void FigureController::printFigureByName(const string
&name) const {
  Figure figure{figureRepository.findByName(name)};

  if (figure.name.empty() || figure.parts.empty()) {
    Logger::error("Figure is empty, cannot prin   );
  } else {         t."
    Logger::info(string(figure));
  }
}
```

LegoClient

```
int main(int argc, char *argv[]) {
  Logger::initialize();
  try {
    if (argc < 2) {
      Logger::error("Error: No figure name provide   );
      exit(1);      d."
    }
    string figureName{argv[1]};
    FigureController().printFigureByName(figureName);
  }
  catch (exception const &e) {
    Logger::error("Client has crashe   , e);
    exit(1);      d."
  }
  Logger::info("Finished.");
  exit(0);
}
```

Figura Lego Propuesta

Partes

UNICORN

description	quantity
brick 2x6 white	1
brick 2x2 white	1
brick 1x1 white	1
brick 1x1 with a stud on the side headlight	2
brick 1x1 with a stud on the side	2
plate 2x2 white	3
plate 2x3 white	1
brick 1x2 with two studs on the side	1
plates 1x2 with one stud on top	2
bracket 1x2-2x2 white	1
brick 1x1 round	4
cone 1x2 white	4
plate 1x1 round black	4
plate 1x2 black	1
cone 1x1 gold	1
plate 1x2 fun color	1
slopes 1x1 fun color	8
eyes 1x1 normal	2

Protocolo Propuesto

- Un servidor de piezas que entra en operación hace broadcast (UDP) en el puerto 7777 con su dirección IP y el nombre de las figuras que contiene. Los servidores intermediarios hacen escucha en el puerto 7777. Al recibir un broadcast de un servidor de piezas, actualiza su tabla interna con la dirección IP y las figuras del servidor nuevo.

- Un servidor intermedio que entra en operación hace broadcast en el puerto 8888, de su dirección IP. Los servidores de piezas hacen escucha en el puerto 8888. Al recibir un broadcast de un servidor intermediario, responde por el puerto 7777 con su dirección IP y el nombre de las figuras que contiene.

- Un cliente se comunica con algún servidor intermedio por HTTPS, esperando respuesta con el HTML de la figura solicitada, o la respuesta adecuada de HTTPS en su defecto.

Protocolo Propuesto

- Al recibir comunicación de un cliente, un servidor intermediario se comunica con el servidor de piezas que contenga la figura solicitada según su tabla interna en el puerto 6666 por TCP. Una vez concretada la conexión, el servidor intermediario envía una hilera de texto que contiene el nombre de la figura solicitada, y el servidor de piezas responde con el HTML adecuado, el cual el servidor intermediario reenvía por HTTPS al cliente.

- Si la figura solicitada no se encuentra en la tabla interna del servidor intermediario, o este no logra concretar conexión con el servidor de piezas, el servidor intermediario retorna la respuesta adecuada de HTTPS al cliente, 404, y actualiza su tabla interna con la ausencia del servidor de piezas del caso.

Gracias!