MÓDULO 1 - EXTRA

SERIALIZAÇÃO

UNIPDS
EDUCAÇÃO

Formatos de Dados

Textuais

- **Posicional** ~1950
- **CSV** ~1970
- **XML** ~1998
- **JSON** ~2006

Binários

- **Serialização Java** ~1995
- **Google Protobuf** ~2001 interno / ~2008 público
- **Apache Avro** ~2009
- **Apache Parquet** ~2013

Um novo objeto

```
public class Pix {

    private Long id;
    private BigDecimal valor;
    private String chaveDestino;

    public Pix(Long id, BigDecimal valor, String chaveDestino) {
        this.id = id;
        this.valor = valor;
        this.chaveDestino = chaveDestino;
    }

    // getters e setters...
}
```

Serializando um objeto Java

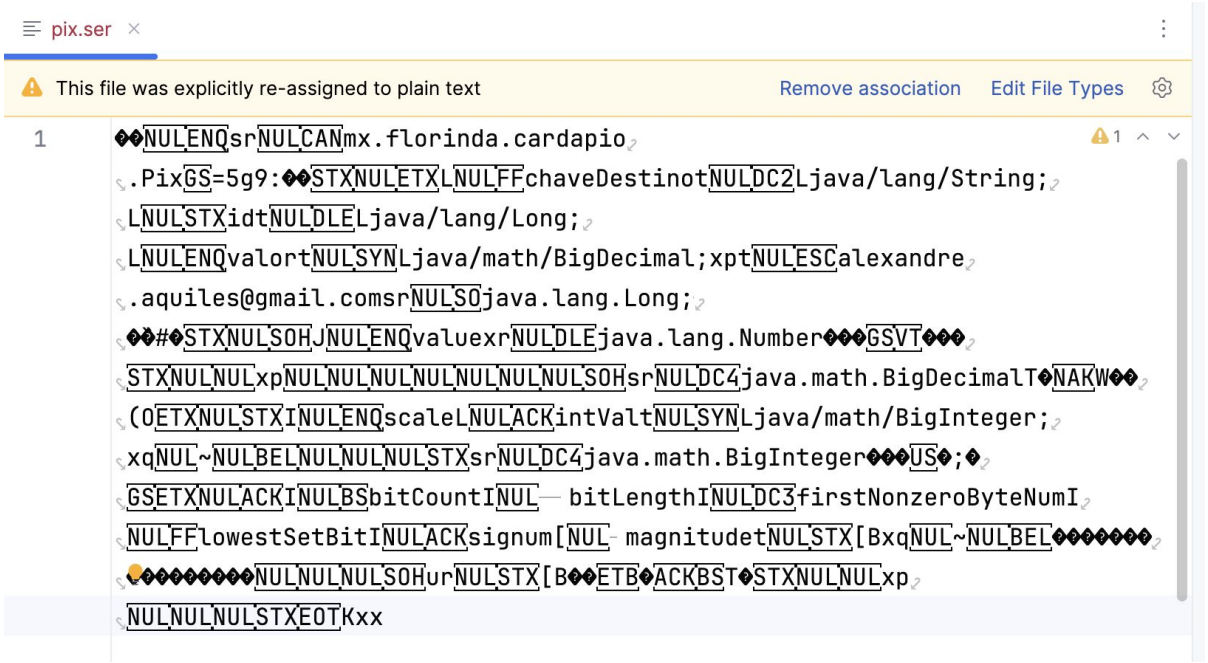
```
public class SerializadorPix {

    public static void main(String[] args) throws Exception {

        var pix = new Pix(1L, new BigDecimal("10.99"),
                                    "alexandre.aquiles@gmail.com");
        FileOutputStream fos = new FileOutputStream("pix.ser");

        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(pix);

    }
}
```


```
Exception in thread "main" java.io.NotSerializableException: mx.florinda.cardapio.Pix
    at java.base/java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1200)
    at java.base/java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:358)
    at mx.florinda.cardapio.SerializadorPix.main(SerializadorPix.java:13)
```

Marcando com Serializable

```
public class Pix implements Serializable {
    //...
}
```

Formato binário



Dá pra entender os tipos e valores dos atributos por que estão codificados em um UTF-8 modificado.

Desserializando um objeto Java

```
public class DesserializadorPix {

    public static void main(String[] args) throws Exception {

        FileInputStream fis = new FileInputStream("pix.ser");

        ObjectInputStream ois = new ObjectInputStream(fis);
        Pix pix = (Pix) ois.readObject();

        System.out.println(pix);
        System.out.println(pix.getChaveDestino());
    }
}
```

Desserializando o objeto

```
mx.florinda.cardapio.Pix@5cb0d902
alexandre.aquiles@gmail.com
```

Faltou o toString

Efeito de mudar a classe

```java
public class Pix {

    //...

    @Override
    public String toString() {
       return "Pix{" +
              "id=" + id +
              ", valor=" + valor +
              ", chaveDestino='" + chaveDestino + '\'' +
              '}';
    }
}
```

Uma versão interna, o `serialVersionUID`, definida a partir dos atributos e métodos é modificada ao mudarmos a classe.

Isso pode tornar a nova classe incompatível com objetos serializados previamente.

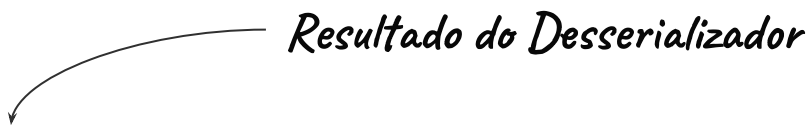
Ao executar o Desserializador novamente

```
Exception in thread "main" java.io.InvalidClassException: mx.florinda.cardapio.Pix; local
class incompatible: stream classdesc serialVersionUID = 2106898918137328632, local class
serialVersionUID = -8121456242453190043
     at java.base/java.io.ObjectStreamClass.initNonProxy(ObjectStreamClass.java:598)
     at java.base/java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:2078)
     at java.base/java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1927)
     ...
```

Controlando a versão manualmente

```java
public class Pix implements Serializable {

    private static final long serialVersionUID = 1L;

    //...
}
```

Podemos adicionar novos métodos como `equals` e `hashCode` e os arquivos serializados anteriormente vão continuar funcionando.




```
Pix{id=1, valor=10.99, chaveDestino='alexandre.aquiles@gmail.com'}
alexandre.aquiles@gmail.com
```

E se adicionarmos novos atributos?

```
public class Pix implements Serializable {

    private static final long serialVersionUID = 1L;

    private Long id;
    private BigDecimal valor;
    private String chaveDestino;
    private Instant dataHora;
    private String mensagem;

    public Pix(Long id, BigDecimal valor, String chaveDestino,
                                    Instant dataHora, String mensagem) {
        this.id = id;
        this.valor = valor;
        this.chaveDestino = chaveDestino;
        this.dataHora = dataHora;
        this.mensagem = mensagem;
    }

    //...
}
```

Adicionando novos atributos, mudando também construtor, toString, equals e hashCode

```
Pix{id=1, valor=10.99, chaveDestino='alexandre.aquiles@gmail.com',
dataHora=null, mensagem='null'}
alexandre.aquiles@gmail.com
```

Ao desserializarmos um arquivo antigo, os novos atributos ficam nulos

Sinalizando uma mudança incompatível

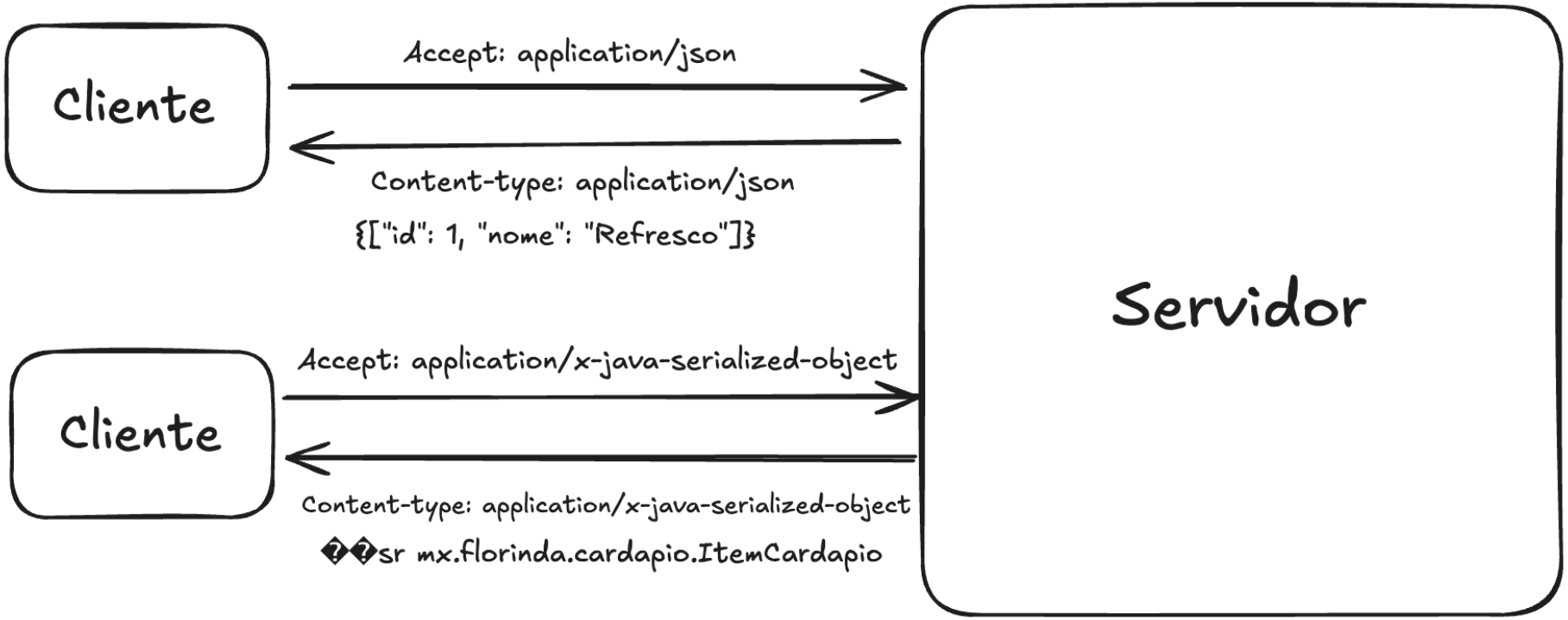
```
public class Pix implements Serializable {

    private static final long serialVersionUID = 2L;

    //...
}
```

Aumentando a versão

Ao desserializarmos um arquivo antigo, uma exceção é lançada

```
Exception in thread "main" java.io.InvalidClassException: mx.florinda.cardapio.Pix; local
class incompatible: stream classdesc serialVersionUID = 1, local class serialVersionUID = 2
    at java.base/java.io.ObjectStreamClass.initNonProxy(ObjectStreamClass.java:598)
    at java.base/java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:2078)
    at java.base/java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1927)
    at java.base/java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:2252)
    ...
```

Content-type Negotiation



Refatorando listagem de itens

```
else if ("GET".equals(method) && "/itens-cardapio".equals(requestURI)) {
   logger.fine("Chamou listagem de itens de cardápio");

   List<ItemCardapio> listaItensCardapio = database.listaItensCardapio();

   String mediaType = "application/json";

   byte[] body;
   Gson gson = new Gson();
   String json = gson.toJson(listaItensCardapio);
   body = json.getBytes(StandardCharsets.UTF_8);

   clientOS.write("HTTP/1.1 200 OK\r\n".getBytes(StandardCharsets.UTF_8));
   clientOS.write(("Content-type: " + mediaType
               + "; charset=UTF-8\r\n\r\n").getBytes(StandardCharsets.UTF_8));
   clientOS.write(body);
   clientOS.flush();
}
```

Usando OutpuStream *com array de bytes*

Verificando media type solicitado

```
String mediaType = "application/json";
for (int i = 1; i < requestLineAndHeadersChunks.length; i++) {
    String header = requestLineAndHeadersChunks[i];
    if (header.contains("Accept")) {
        logger.info(header);
        mediaType = header.replace("Accept: ", "");
    }
}
```

Extrai Accept se estiver definido

Serializando itens

```
byte[] body;
if ("application/x-java-serialized-object".equals(mediaType)) {
    logger.info("Enviando objeto java serializado");
    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(bos);
    oos.writeObject(listaItensCardapio);
    body = bos.toByteArray();
} else {
    Gson gson = new Gson();
    String json = gson.toJson(listaItensCardapio);
    body = json.getBytes(StandardCharsets.UTF_8);
}

public record ItemCardapio(Long id, String nome, String descricao,
                           CategoriaCardapio categoria, BigDecimal preco,
                           BigDecimal precoPromocional) implements Serializable {
}
```

Serializando a lista de itens

Testando com HttpClient

```
URI uri = URI.create("http://localhost:8000/itens-cardapio");

try(HttpClient httpClient = HttpClient.newHttpClient()) {
   HttpRequest httpRequest = HttpRequest.newBuilder(uri)
           .header("Accept", "application/x-java-serialized-object")
           .build();
   HttpResponse<byte[]> httpResponse = httpClient.send(httpRequest,
                                                       HttpResponse.BodyHandlers.ofByteArray());
   int statusCode = httpResponse.statusCode();
   byte[] body = httpResponse.body();
   System.out.println(statusCode);
   System.out.println(body);

   ByteArrayInputStream bis = new ByteArrayInputStream(body);
   ObjectInputStream ois = new ObjectInputStream(bis);
   List<ItemCardapio> itens = (List<ItemCardapio>) ois.readObject();
   itens.forEach(System.out::println);
}
```

Define cabeçalho Accept

Body Handler binário

Desserialização dos itens

Testando com cURL

```
$ curl -v -H 'Accept: application/x-java-serialized-object'
                                  localhost:8000/itens-cardapio --output lista.ser

< HTTP/1.1 200 OK
< Content-type: application/x-java-serialized-object; charset=UTF-8
<
* no chunk, no close, no size. Assume close to signal end
{ [2135 bytes data]
100  2135    0  2135    0     0   3375      0 --:--:-- --:--:-- --:--:--  3378
* Closing connection

$ cat lista.ser
??srjava.util.ArrayListx????a?Isizexsr!mx.florinda.cardapio.ItemCardapioL
categoriat5Lmx/florinda/cardapio/ItemCardapio$CategoriaCardapio;L
descricaotLjava/lang/String;LidtLjava/lang/Long;Lnomeq~LprecotLjava/math/BigDecimal;LprecoProm
ocionalq~xp~r3mx.florinda.cardapio.ItemCardapio$CategoriaCardapioxrjava.lang.EnumxptBEBIDASt?S
uco de limão que parece de tamarindo e tem gosto de groselha.srjava.lang.Long;??
#?Jvaluexrjava.lang.Number???
                                                  ???xptRefresco do
```