TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Xeración automática de paneis de información personalizados con contido multimedia e deseño enriquecido

Estudante: Brais García Brenlla

Dirección: José Carlos Dafonte Vázquez

Ángel Gómez García

A Coruña, xuño de 2022.

A miña avoa, que é tan preciosa como o seu nome indica

Agradecementos

Primeiro, quero darlle as grazas a meus pais, miña irmá e miña avoa por apoiarme sempre e axudarme a chegar ata aquí, grazas por confiar en min desde sempre.

Tamén dar as grazas a toda a xente que me acompañou durante a carreira e incluso dende antes. Grazas David, Roi, Miguel, Diego e Javi por aturarme de compañeiro de piso; Ángel, Dani, Juan e Jaime por facer máis levadeira as épocas de traballos e exames; e a Miguel Ángel e Jorge por estar comigo incluso estando lonxe.

Finalmente dar grazas a todo os profesores que me ensinaron todo o que sei, entre os que se atopan José Carlos Dafonte Vázquez e Ángel Gómez García, os directores deste TFG; así como Mónica e Manuel, as primeiras persoas en ensinarme o que era a informática.

Resumo

Este traballo de fin de grao busca realizar un servizo de organización de información recibida a través de XML (Linguaxe de Marcado Extensible) para o seu posterior uso en taboleiros web de fácil acceso desde outros dispositivos con acceso á mesma rede.

Trátase de un sistema altamente adaptable a múltiples escenarios nos que faga falta unha visualización a tempo real de información, permitindo cambiar o formato dos datos almacenados ou a organización da información que se mostra como saída para o usuario para adaptalos como sexa necesario. Debido a esta gran plasticidade neste TFG centrarémonos no caso concreto dun portal de noticias que traballa cunha información e visualización simplificada.

A información que xestionará o servizo dependerá totalmente do contexto de emprego, pero non só constará de datos sobre o que se desexa visualizar, tamén poderá incluír datos extra para modificar como se visualiza, como poden ser a cor de fondo, tipo de letra, posición dentro da páxina,etc.

Dentro do sistema podemos atopar dous tipos de módulos, un módulo fixo, que sempre é igual para todos os contextos, e dous módulos adaptables, que dependerá do caso concreto no que se queira usar o servizo. Estes segundos son: o analizador léxico-sintáctico xunto coa clase que describe a información que este le do XML, e a parte de deseño web, pois o servizo funciona a modo de API (Application Programming Interface) , o cal nos permite poder adaptar a visualización ao gusto do cliente que implemente o servizo, independentemente da xestión dos datos.

Abstract

This project seeks to perform a service for organizing information received through XML (Extensible Markup Language) for use on easily accessible online news boards from other devices with access to the same network.

It is a highly adaptable system for scenarios that requires a real-time display of information, allowing changes like using different formats for the stored data or have different organizations for the information that is displayed as output to the user. Due to this great plasticity in this TFG we will focus on the specific case of a news portal that works with simplified information and visualization.

The information that will manage the service will depend entirely on the context of employment, but will not only consist of data on what you want to view, it may also include extra data to change how it is displayed, such as background color, font, position inside the page ...

Within the system we can find two parts, a fixed part, which is always the same for all contexts, and an adaptable part, which will depend on the specific case in which you want to use the service. Within this second, there are two different parts: the Lexical-Syntactic Analyzer along with the class that describes the information it reads from XML, and the web design part, as the service works as an API (Application Programming Interface) it allows us to adapt the visualization to the taste of the customer who implements the service, regardless of data management.

Palabras chave:

- Java
- Jflex
- Cup
- Protocolo HTTP
- Analizador Léxico-Sintáctico
- Taboleiro de Información

Keywords:

- Java
- Jflex
- Cup
- HTTP protocol
- Lexical-Syntactic Analyzer
- News boards

Índice Xeral

Índice de Figuras

Índice de Táboas

Capítulo 1

Introdución

Neste primeiro capítulo faremos fincapé no porqué da realización deste proxecto expoñendo os estímulos que deron lugar ao xurdimento desta idea, dando múltiples exemplos de por que este traballo pode ser de gran utilidade e comparándoo con outras alternativas. Tamén falaremos da estrutura desta memoria explicando os contidos de cada parte.

1.1 Contexto

Para a sociedade actual que vive nun mundo hiperconectado e globalizado a información en tempo real é un dos bens máis prezados e comúns, chegando a ser unha necesidade indispensable en moitos escenarios: noticias informativas en directo, anuncios dentro de organizacións ou empresas, últimas ofertas en centros comerciais, estado das liñas de transporte público e tráfico para estacións, carteis publicitarios e moitos máis casos.

Moitas veces esta necesidade crea excesos de información, os cales causan problemas tanto a creadores coma a usuarios. No que respecta aos usuarios non é de estrañar que esta abrumante cantidade de estímulos cause unha confusión innecesaria, en moitos casos empeorada por ser difícil de comprender por mor dunha mala organización. Como contraparte, os creadores deben gastar unha cantidade inxente de recursos para poder procesar e organizar esa información de forma correcta, provocando que moitas veces non se analice e ordene como é debido por falta de capacidade.

É por isto que vivimos rodeados de medios de comunicación que nos serven para recibir información de infinidade de fontes e formatos que nos manteñen conectados as 24 horas do día, dende os telexornais da televisión ás mensaxes que podemos ler en calquera rede social, pasando por paneis informativos sobre os estados do tráfico nunha cidade. Trátase de algo ao que estamos tan acostumados que semella sinxela, pero non o é. O tratamento da información é algo tan importante á par que complexo que existen ramas de múltiples disciplinas adicadas solo a isto. Só en informática podemos atopar profesionais que se adican só á creación de bases

de datos, creación de servizos de envío e recepción de información ou deseño web, buscando crear sistemas que permitan almacenar e mostrar información da maneira ordenada e cómoda. Incluso existen carreiras como Comunicación Audiovisual [1], que se centra no tratamento e creación de contido audiovisual, ou Ciencia e Enxeñaría de Datos [2], onde se estuda como tratar os datos de forma correcta.

No intento por mellorar a situación fíxose moi común o uso dos contidos enriquecidos. Son contidos creados por programas que decoran e organizan a información dando lugar a produtos moito máis amigables de cara ao lector. Algúns dos exemplos máis comúns de creadores deste tipo de contidos son Canva [3] ou Blogger [4], sistemas que permiten ao usuario crear páxinas ordenadas e personalizadas nas que expoñer información de xeito ordenado, usando multitude de recursos e ferramentas audiovisuais.

Motivados pola mesma problemática xestionar enormes cantidades de datos, é normal que o uso de sistemas informáticos para tratar a información sexa cada vez máis importante e máis demandado por grandes organizacións. Neste contexto é de onde parte a idea deste proxecto, pois en materia de noticias é normal a creación de recompilacións das novas diarias ou semanais en páxinas resume ou vídeos, normalmente maquetados ou editados manualmente por un grupo de deseñadores. A creación dun sistema que automatice total ou parcialmente este proceso supón por unha parte un alivio a nivel económico e por outra unha redución de carga de traballo, que supoñerían un gran beneficio para a organización, proporcionando ao mesmo tempo un mecanismo que permita transmitir a información empregada de forma sinxela para calquera portal que o precise mediante un provedor de noticias que pode ser adaptado cando sexa preciso para adaptarse perfectamente ao seu contexto. Adicionalmente esta información acompañarase de parámetros de estilo que o portal poderá empregar de forma opcional, pero que simplifican a visualización e categorización.

Por todo isto xorde a idea de crear un sistema capaz de analizar XML entrantes que traerán información como título, categoría, autor, etc.; así como un conxunto de campos que conteñan información sobre o estilo recomendado para a noticia como cor de fondo, cor da letra, tipo de letra, etc; de tal forma que manteña actualizado un taboleiro de noticias en tempo real con contido enriquecido e que poida ser visualizado dende calquera buscador con acceso ao servizo, permitindo a súa implantación con facilidade en multitude de contornas.

Trátase dun sistema con gran número de utilidades que aínda que foi inicialmente pensado para páxinas web de noticias tamén pode ser utilizado en moitos outros casos, coma un resumo das compras e ventas dunha empresa, un taboleiro de actividades nun centro educativo ou un calendario de eventos dunha organización. Ademais, ao tratarse de contido enriquecido mediante os parámetros de estilo, usando por exemplo códigos de cores ou diversos tipos de letra, facilitaráselle ao usuario atopar aquela información que precise da forma máis fácil posible e aumentando a comprensión da información para posibles novos usuarios.

1.2 Estrutura da memoria

Para facilitar a comprensión e lectura desta memoria estructurouse en 8 capítulos, os cales comentaremos a continuación:

- **Capítulo 1. Introdución**: Capítulo actual no que se pon ao lector en contexto sobre o proxecto que se explica ao longo desta memoria e a estrutura desta.
- **Capítulo 2. Motivacións e obxectivos**: Neste capítulo explícanse os diversos obxectivos que se propuxeron, así como por que se escolleron.
- **Capítulo 3. Estado da arte**: Neste capítulo analízanse diversos métodos e ferramentas que se usan actualmente para traballar con información.
- **Capítulo 4. Ferramentas empregadas**: Neste capítulo lístanse todas as ferramentas e tecnoloxías empregadas para a realización do proxecto.
- **Capítulo 5. Metodoloxía**: Neste capítulo farase unha introdución á metodoloxía empregada.
- **Capítulo 6. Planificación e estimación de custos**: Neste capítulo explicarase a planificación das tarefas realizadas ao longo do proxecto facendo unha análise dos custos estimados.
- **Capítulo 7. Desenvolvemento do proxecto**: Neste capítulo explicarase detalladamente cada un dos incrementos levados a cabo no proxecto.
- **Capítulo 8. Conclusións e liñas futuras**: Finalmente farase unha análise da aplicación creada en comparación cos obxectivos iniciais e o aprendido durante o grao, e propoñeranse novas metas de cara a futuras aproximacións.

Capítulo 2

Motivacións e obxectivos

UNHA vez visto o estado da arte que existe podemos analizar o entorno para decidir cal é o tipo de sistema que queremos crear, así como todos as características que cremos máis necesarias que este conteña.

2.1 Motivacións

Tras estudar todas as diversas alternativas existentes para tratar con información (explicadas en detalle no capítulo 3) podemos observar que actualmente úsanse tres formas diferentes de traballar con esta:

- A forma tradicional, coa axuda de creadores de contido enriquecido, que se basean en ter profesionais analizando e categorizando a información manualmente. Consegue resultados de calidade, pero ningunha parte do traballo está automatizada e ten un gasto de recursos e tempo que non sempre é posible usar.
- Sistemas que adquiren información doutras fontes. Son o caso dos sistemas RSS e os buscadores de noticias automatizados. Trátanse de sistemas dependentes que poden ter problemas legais por dereitos de autor e que non poden ser empregados sen conexión a outros servizos.
- Sistemas privados. Son sistemas feitos especificamente para unha organización ou empresa. O seu principal problemas reside en que ao ser de carácter privado o máis posible é que non poidan ser usados fóra do ámbito para o que foron creados, carecen de adaptabilidade, e ademais non podemos saber como se organiza a información dentro do sistema, polo que é imposible saber se realmente automatiza algunha parte do traballo.

En conclusión, non existe, polo menos de forma pública, ningún sistema de información en tempo real, autoorganizable, autosuficiente e adaptable a múltiples contextos ao mesmo tempo; dándonos isto unha idea das características que interesaría que o noso sistema tivese.

2.2 Obxectivos

Unha vez revisadas as opcións existentes na actualidade chegamos á conclusión de que non existe ningún sistema capaz de cumprir exactamente os nosos requirimentos. Debido a isto decidiuse desenvolver un xestor de información con visualización enriquecida. Conseguirase mediante un sistema capaz de recibir XMLs con información mediante unha API que os organiza e os mantén accesibles. Este sistema permitirá solicitar os datos en calquera momento, permitindo a visualización desta información por pantalla dun xeito ordenado e actualizado en tempo real en calquera dispositivo con acceso á rede, incluso nos de menor potencia. Para conseguilo podemos establecer catro funcionalidades imprescindibles para levar a cabo no traballo:

- **Deseño da estructura dos datos**: Deseño da estrutura dos XML de entrada xunto coa súa gramática e os JSON de saída, de tal forma que nos permita a súa análise e creación. Esta parte é dependente do contexto, pois non serve o mesmo XML ou JSON para un noticieiro que para unha empresa.
- **Análise de información**: Implementación dun analizador léxico-sintáctico que nos permita analizar os XML de entrada para xerar a saída coa información ordenada.
- **Servizos en liña**: Creación dun sistema capaz de recibir os XML de entrada e enviar a información de saída a través da rede.
- **Deseño da visualización final**: Deseño de como se mostrará ao usuario toda a información que foi clasificada. Dependerá do contexto de utilización, pois debe adaptarse aos deseños existentes do cliente.

Para poder cumprir todos os obxectivos deste proxecto o sistema debe presentar as seguintes características:

- **Tempo real**: o sistema debe ser capaz de recibir nova información en todo momento e que o cliente poida visualizar a mais recente de forma simple sen necesidade de recargar a páxina.
- **Autoorganizable**: a páxina debe xerarse de forma semiautomática e todas as novas deben formatearse de forma automática a partir da información recibida.
- **Autosuficiente**: o sistema debe ser capaz de funcionar nunha contorna na que non teña acceso a outros servizos.

- **Adaptable**: independentemente da finalidade que se lle queira dar ao sistema este debe precisar unha edición mínima para adaptarse a contorna. Por suposto contornas máis diferentes requerirán cambios de maior magnitude.

Capítulo 3

Estado da arte

PARA entender mellor os obxectivos que queremos acadar neste proxecto e poder facer un sistema o máis útil posible analizaremos diversas formas e ferramentas que traballan con información. Desta forma poderemos comprobar os puntos fortes e os punto fracos de cada unha para telos en conta neste traballo.

3.1 Tradicional

Trátase da forma máis antiga, pois é a única que existía antes dos sistemas informáticos. É unha alternativa con grandes beneficios, pero altos custos; pois basease en ter un grupo de profesionais da materia realizando a análise, clasificación e enriquecemento de forma manual. O punto máis determinante deste método é a calidade dos profesionais, dado que se teñen maior coñecementos e experiencia obterán un produto de maior calidade; pero ao mesmo tempo estes factores tamén incrementan o valor destes recursos, polo que a empresa debe decidir entre maior calidade ou reducir custos.

Este método é moi usado sobre todo en usos nos que a calidade é unha necesidade crítica; un exemplo deste caso son os programas en directo, para o correcto enriquecimento da información existe un grupo de postprodución encargado de engadir imaxes, textos ou sons á emisión cando esta o requira. O uso desta alternativa neste contexto é moi útil pola adaptabilidade que ten, sumado á que equipos de xente moi experimentada consegue resultados de moi alta calidade.

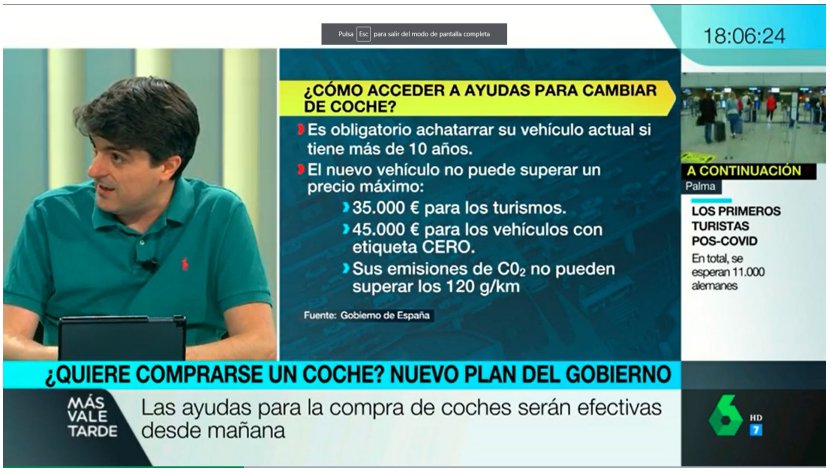


Figura 3.1: Exemplo de maquetado en noticiario en directo

3.2 Buscadores de noticias automatizados

Outras alternativas que si son implementadas por sistemas no ámbito das noticias son os buscadores de noticias automatizados. Son sistemas que recollen noticias de diversas fontes para analizalas e clasificalas, para despois ofrecerlle esa información aos usuarios de forma ordenada. Trátanse de sistemas moi amigables de cara ao usuario, pois xunta as noticias de multitude de fontes nun único lugar.

Nalgúns casos, por exemplo Google News e Google Play Quiosco, ambas de Google; o sistema non só ordena as noticias, senón que tamén analiza os gustos do usuario de tal forma que poida recomendar aquelas noticias que cre que poden ser de interese para o lector.

Como xa se dixo, este sistema é moi cómodo de cara aos usuarios, pois teñen toda a información que queren recollida nun único lugar e incluso organizada polos seus gustos, pero tamén ten inconvenientes. O sistema emprega a información doutras fontes, dando lugar a que a aplicación sexa totalmente dependente delas. De feito, esta dependencia causou o peche de Google News en España dende o 2014 ata o 22 de xuño de 2022, debido a que a lei de propiedade intelectual [5] de España obrigáballe a pagar aos editores das noticias; o cal cambiou o 2 de decembro de 2021, pois a modificación da susodita lei [6] permite negociar coas fontes os pagos a realizar.

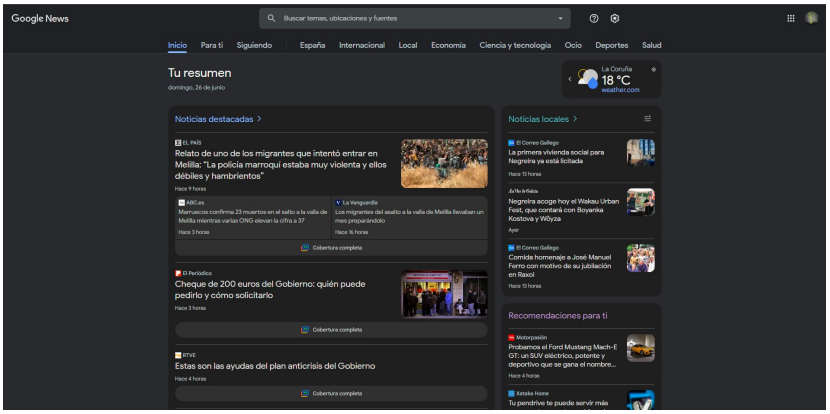


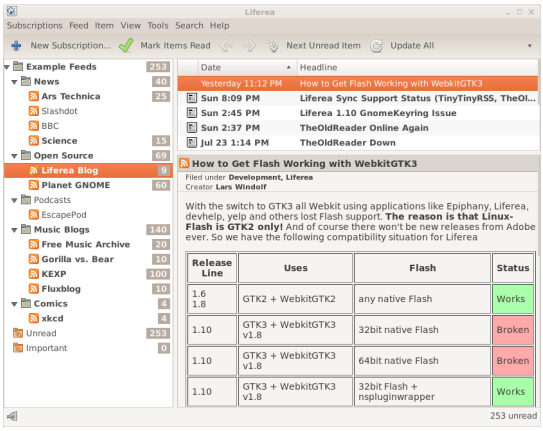
Figura 3.2: Páxina inicial de Google News

3.3 Sistemas RSS

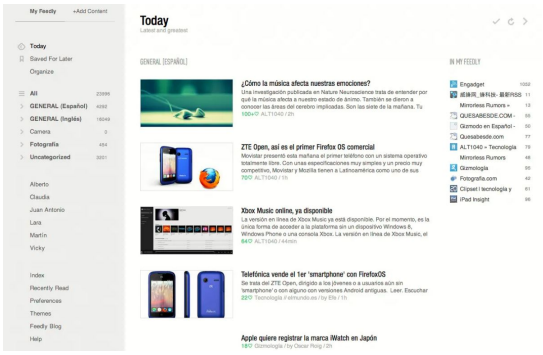
Moi similares aos buscadores de noticias automatizados son os sitemas RSS (*Really Simple Syndication*) [7], os cales permiten aos usuarios subscribirse a páxinas de información, de tal xeito que a aplicación recolla as novas de todas as fontes ás que estean subscritos. É un sistema útil para os usuarios, pois estes teñen un gran control sobre a orixe da información, ademais que unifica nun único lugar todas a información que o usuario quere.

A pesar de que o sistema consta destes beneficios tamén hai que ter en conta os seus puntos negativos; ao igual que no caso anterios, depende doutras fontes de información, o cal pode causar problemas; e ademais, a necesidade por parte do usuario de subcribirse ás diversas fontes provoca que non se poidan recuperar novas cuxa orixe sexa descoñecida polo lector.

Algúns exemplos de sistemas que usen RSS son Liferea [8] e Feedly [9]; o primeiro é unha aplicación de escritorio, mentres que o segundo úsase mediante o navegador.



(a) Pantalla principal de Liferea



(b) Páxina principal de Feedly

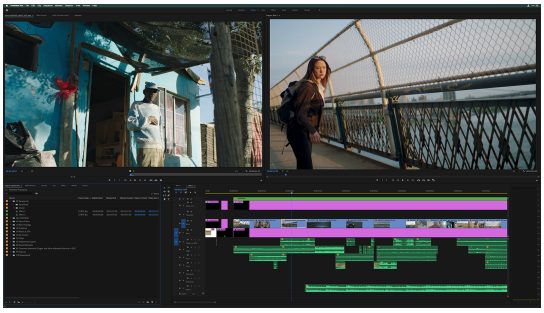
Figura 3.3: Sistemas RSS

3.4 Creadores de contido enriquecido

Unhas ferramentas totalmente diferentes ás anteriores son os creadores de contido enriquecido. Dentro deste tipo de ferramentas podemos atopar sistemas de edición de vídeo como Adobe Premier [10] ou calquera outro editor de vídeo, e sistemas de xeración de páxinas ou taboleiros como Showscreen [11] ou calquera xestor de contido.

Con esta ferramentas pódense crear recursos como vídeos con subtítulos, imaxes, sons extra, etc; que foron incluídos para a mellor comprensión dos contidos; ou paneis que fan uso de cores, tipos de letra, imaxes, etc; para que sexa fácil atopar a información de interese para cada lector.

Estes programas permiten ao usuario crear contido enriquecido utilizando diversas ferramentas audiovisuais que facilitan enormemente o traballo dos deseñadores, pero seguen precisando que alguén cree ese contido, polo que se trata dunha ferramenta que simplifica a técnica tradicional, pero non sistematiza nada.



(a) Pantalla principal de Adobe Premier Pro

(b) Exemplo de taboleiro creado con Showscreen

Figura 3.4: Creadores de contido enriquecido

3.5 Sistemas privados

Moitas empresas e organizacións contan con aplicacións feitas especificamente para tratar coa información do seu ámbito e orixe concreto, como poden ser as aplicacións que usan *La Voz de Galicia*, *La Moncloa* (Web oficial do presidente do Goberno e do Consello de Ministros) ou a páxina da propia UDC.

Este sistemas son creados a medida para o ámbito no que son usados, polo que usualmente a súa calidade é moi alta, axustándose perfectamente as necesidades do usuario final.

O principal inconveniente destes sistemas é que deben ser deseñados e creados especialmente para cada situación, o cal xera un gasto de recursos e tempo importante para a empresa ou organización. Ademais, ao tratarse de sistemas privados é imposible saber como é o seu funcionamento interno.

(a) Páxina principal de *La Voz de Galicia*

(b) Páxina principal de *La Moncloa*

Figura 3.5: Creadores de contido enriquecido

Capítulo 4

Ferramentas empregadas

Este capítulo centrarase en comentar todas as ferramentas e tecnoloxías usadas para este proxecto falando da súa utilidade así como as razóns da súa escolla.

4.1 Java

Java [12] é unha linguaxe de progamación ampliamente usada que se basea na programación obxectual. Está inspirada en C e C++, pero simplificando as partes de máis baixo nivel, como por exemplo a xestión de punteiros.

Foi escollida para este proxecto pola súa gran cantidade de bibliotecas e manuais, que facilitan o desenvolvemento de novos programas. Ademais é unha linguaxe moi usada en canto á creación de servizos pola súa simpleza á hora de crear conexións e o seu funcionamento basado nunha máquina virtual, que facilita o seu uso na maioría dos dispositivos.

Na realización deste proxecto cabe destacar a utilización de dúas bibliotecas: HttpServer e Common-Text, das cales falaremos a continuación. Tamén se usou a biblioteca java_cup.runtime, necesaria para o uso de CUP (Sección 4.3) dentro de java.

Figura 4.1: Logo de Java

4.1.1 HttpServer

HttpServer [13] é unha das bibliotecas incluídas en Java que permite a creación dun servizo HTTP (*Protocolo de transferencia de hipertexto*) mediante a implementacion da interfaz *Http-*

Handler . Esta clase incluirá unha función *handle* que se encargará de xestionar as peticións que o servizo reciba sen necesidade de controlar a conexión co cliente.

4.1.2 Common-Text

Common-Text [14] é unha biblioteca creada por Apache Commons que inclúe diversas funcións para traballar con cadeas de texto. No caso do noso proxecto úsase polas utilidades aportadas por *StringEscapeUtils*, con funcións que nos permiten formatear as cadeas de texto de tal forma que sexan seguras para usar en protocolos como HTML(sección 4.4) ou JSON(sección 4.8). Débese ter en conta que esta biblioteca depende de *Commons-Lang* [15], tamén de Apache Commons.

4.2 JFlex

JFlex [16] é un xerador de analizadores léxicos para Java basado en Flex [17]. Con el pódense xerar clases capaces de analizar lexicamente cadeas de texto dentro de java de forma rápida e eficaz usando autómatas finitos deterministas.

Un xerador de analizadores léxico como JFlex toma como entrada un conxunto de expresións regulares e accións correspondentes. Xera un programa (un *lexer*) que le a entrada, comparaa coas expresións regulares do ficheiro de especificacións e executa a acción correspondente se unha expresión regular coincide. Os lexers adoitan ser o primeiro paso *front-end* nos compiladores, facendo coincidir palabras clave, comentarios, operadores, etc. e xerar un fluxo de tokens de entrada para utilizar para os *parsers*. Os lexers tamén se poden usar para moitos outros propósitos.

Pódese usar directamente para certas tarefas, pero o máis habitual é usalo conxuntamente cun analizador sintáctico. Os analizadores sintácticos recomendados polos creadores de JFlex son CUP (sección 4.3), creado por Scott Hudson, ou a modificación de Berkeley Yacc para Java denominada BYacc/J [18], creada por Bob Jamison. Neste proxecto decantámonos pola primeira das dúas opcións.

4.3 CUP

CUP (*Construction of Useful Parsers*) [19] é un xerador de analizadores sintácticos para Java. Este baséase na creación dun analizador LALR(1), que permite analizar unha estrutura seguindo unha regras que conformen unha gramática válida usando un *token* de anticipación. Cumpre o mesmo papel que o programa amplamente utilizado YACC (*Yet Another Compiler-Compiler*) e de feito ofrece a maioría das características de YACC. Non obstante, CUP está

escrito en Java, usa especificacións incluíndo código Java incorporado e produce analizadores que se implementan en Java.

Figura 4.2: Logo de CUP

4.4 HTML

HTML(*HyperText Markup Language*) [20, 21] é un estándar de marcado ampliamente utilizado no deseño de páxinas web. Actualmente está controlado por W3C(*World Wide Web Consortium*), a organización encargada de supervisar todas as tecnoloxías ligadas coa rede.

Figura 4.3: Logo de HTML

4.5 CSS

CSS(*Cascading Style Sheets*) [21] é unha linguaxe de estilos que permite controlar o aspecto de HTML na súa visualización no navegador. Ao igual que HTML, esta tamén está controlado por W3C.

Figura 4.4: Logo de CSS

4.6 JavaScript

Javascript [22] é unha linguaxe de programación lixeira con orientación a obxectos que está pensada para o seu uso en navegadores conxuntamente con HTML para poder crear paxinas web con partes executables. É unha das linguaxes base da Open Web e posúe unha especificación estandarizada por parte da W3C.

Dentro deste proxecto será a tecnoloxía empregada para permitir que a páxina web que o usuario visualiza se poida actualizar en tempo real.

Figura 4.5: Logo de JavaScript

4.7 XML

XML(*eXtensible Markup Language*) [23] é unha linguaxe de etiquetaxe que busca dar formato aos datos durante o seu intercambio. Trátase dunha linguaxe regulada pola W3C.

4.8 JSON

JSON(*JavaScript Object Notation*) [24] é un formato de intercambio de datos. Trátase dun formato que busca ser o máis simple posible para a súa utilización e creación, pero ao mesmo tempo, a pesar de non ter nada que ver con JavaScript(sección 4.6), os convenios que teñen entre eles permitén un uso conxunto case perfecto.

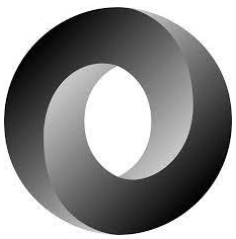


Figura 4.6: Logo de JSON

4.9 Eclipse

Eclipse [25] é un IDE(*Integrated Development Environment*) que soporta a programación, compilación e execución en Java e C. Está dispoñible para os principais sistemas operativos: Windows, GNU/Linux e Mac OS.

Eclipse aporta un entorno de desenvolvemento que facilita ao programador o seu traballo á hora de crear código. Neste editor de código fonte temos acceso a múltiples funcionalidades útiles como poder navegar pola xerarquía de arquivos, autocompletar código ou realizar depuración dunha forma sinxela.

Figura 4.7: Logo de Eclipse

4.10 Postman

Postman [26] é un sistema que permite ao usuario realizar peticións HTTP de forma sinxela. Trátase dunha aplicación moi útil nas etapas de probas do sistema, pois permite realizar peticións personalizadas, gardar coleccións de peticións que se poden usar múltiples veces, visualizar de forma directa as respostas recibidas, etc. Está dispoñible para os principais sistemas operativos: Windows, GNU/Linux e Mac OS; ou tamén se pode usar directamente desde un navegador.

Figura 4.8: Logo de Postman

4.11 GitHub

GitHub [27] é un sistema de versións amplamente utilizado. Trátase dun sistema actualmente propiedade de Microsoft usando Git. Decidiuse usar esta ferramenta neste proxecto

por ser unha alternativa gratuíta e coa que se estaba familiarizado.

Figura 4.9: Logo de GitHub

4.12 LaTeX

LaTeX [28] é un sistema de composición de textos útil para a creación de arquivos PDF de alta calidade tipográfica. Este sistema foi o utilizado durante a redación desta mesma memoria.

Figura 4.10: Logo de LaTeX

4.13 OverLeaf

OverLeaf [29] é un editor en liña de LaTeX que permite ao usuario ver en tempo real como se visualizará o documento final. Trátase dunha ferramente de gran utilidade, pois permite o traballo colaborativo nun mesmo proxecto e a visualización e exportación do ficheiro PDF final sen necesidade de instalar ningún software.

Figura 4.11: Logo de Overleaf

Capítulo 5

Metodoloxía

Unha metodoloxía é un conxunto de técnicas e métodos que se usarán ao longo do desenvolvemento dun proxecto, ligándoos ao ciclo de vida deste. Estas metodoloxías son as encargadas de marcar as pautas que encauzarán que o proxecto cumpra os seus obxectivos. Tendo en conta isto, non cabe dúbida que escoller a metodoloxía correcta á hora de realizar un proxecto é un paso clave, pois o uso de dúas metodoloxías diferentes pode dar lugar a resultados totalmente distintos e causar gastos ou perdas de tempo innecesarios.

Para este traballo deciduse usar unha metodoloxía áxil, pois os seus beneficios ante a metodoloxías tradicionais son moi amplos. Algúns destes principais beneficios son:

- Maior facilidade de seguimento e concentración no traballo, pois a división en partes permite centrarse en seccións máis reducidas do proxecto sen verse influído polas demais.
- O proceso de desenvolvemento en incrementos permite unha maior flexibilidade á hora de aceptar cambios nos requisitos de análise, ademais que simplifica o proceso de probas.
- Permite ter un seguimento por parte do cliente, pois ao final de cada incremento poderá ver parte do produto e dar as súas opinións.
- Grazas ao tamaño máis reducido dos incrementos evítanse problemas de maior tamaño, minimizando sobrecustos e perdas de tempo.

5.1 Metodoloxía: *Scrum*

Actualmente unha das metodoloxías áxiles máis populares é Scrum[30]. Trátase dunha metodoloxía apta para problemas complexos e adaptativos, intentando resolvelos coa maior calidade posible. A pesar de ser unha metodoloxía de fácil comprensión, o seu dominio non é

sinxelo, pois a diferenza doutras metodoloxías que centran o seu peso no xefe de equipo esta require tamén unhas habilidades de autoorganización e xestión da relación cos compañeiros por parte de todos os membros do equipo. Para comprender correctamente o funcionamento de Scrum debemos falar dos seus tres conceptos esenciais: roles, eventos e artefactos.

5.1.1 Roles

Dentro de Scrum podemos atopar tres roles ben diferenciados:

- **Product Owner**: encárgase do *Product Backlog*, é o encargado de recopilar e organizar as ideas e opinión do cliente.
- **Scrum Master**: o seu principal traballo é supervisar o correcto funcionamento do equipo á hora de seguir o Scrum, así como encargarse de xestionar as relacións externas para que non interfiran na metodoloxía. Tamén axuda ao *Product Owner* na xestión do *Product Backlog*. Pódese dicir que é o encargado de evitar todos os obstáculos posibles e axudar nos procesos que o precisen.
- **Development Team**: fórmase por un pequeno grupo de profesionais que de xeito autónomo se organizan para realizar as tarefas do *Product Backlog* de forma ordenada para crear os diversos *Sprints*.

5.1.2 Eventos

Durante o transcurso do proxecto o equipo levará a cabo varios eventos estipulados por Scrum, buscando así crear un ambiente regular que simplifique o traballo evitando a organización de reunións e xuntanzas non definidas. Unha das características máis importantes dos eventos é a existencia dun *time-box*(un limite de tempo) para a duración destes. Os diversos eventos son:

- **Sprint**: trátase do evento máis importante de Scrum. É un bloque de tempo no que se desenvolve un dos incrementos do produto. Por regra xeral soen ser todos da mesma duración ao longo do proxecto.
- **Sprint Planning**: é a reunion inicial de cada Sprint. Nela analízase o *Product Backlog* para determinar os obxectivos do *Sprint* en cuestión. Non debe superar as oito horas de duración.
- **Daily Scrum**: unha reunión diaria de quince minutos na que os membros do equipo explican o feito o día anterior e o que farán ese día para a súa sincronización. Normalmente faise sempre no mesmo lugar e hora para simplificar.

- **Sprint Review**: trátase da reunión a realizar na entrega de cada *Sprint*. É de carácter máis informal que o resto, pois o seu obxectivo é ver os resultados obtidos e recibir a opinión dos diversos membros do equipo.
- **Sprint Retrospective**: realízase entre o *Sprint Review* e o *Sprint Planning*. Nas tres horas de duración desta reunión o equipo fará un balance do último *Sprint* e identificará posibles melloras para o seguinte.

5.1.3 Artefactos

Os artefactos de Scrum son tarefas ou valores en diferentes formas, sendo estas útiles para aumentar as posibilidades de inspección e adaptación. Existen os seguintes artefactos:

- **Product Backlog**: trátase dunha lista ordenada por prioridade dos diversos requisitos do sistema. Nela o *Product Owner* encargase de priorizar os requisitos que poidan aportar máis valor ao proxecto e detallalos de tal forma que o equipo os poida comprender e desenvolver da mellor forma posible.
- **Sprint Backlog**: é o conxunto de requisitos do *Product Backlog* que foron escollidos polo equipo para ser implementados no *Sprint*. Corresponde ao que o equipo pensa que consolida un incremento finalizado con un fragmento de código funcional.
- **Incremento**: é o conxunto de todos os requisitos completados duranto o *Sprint* actual e o valor de todo os incrementos anteriores. Debe ser un produto totalmente funcional.
- **Sprint Burndown Charts**: trátase dun gráfico que mostra como se van completando os diversos obxectivos do proxecto ao longo do tempo.

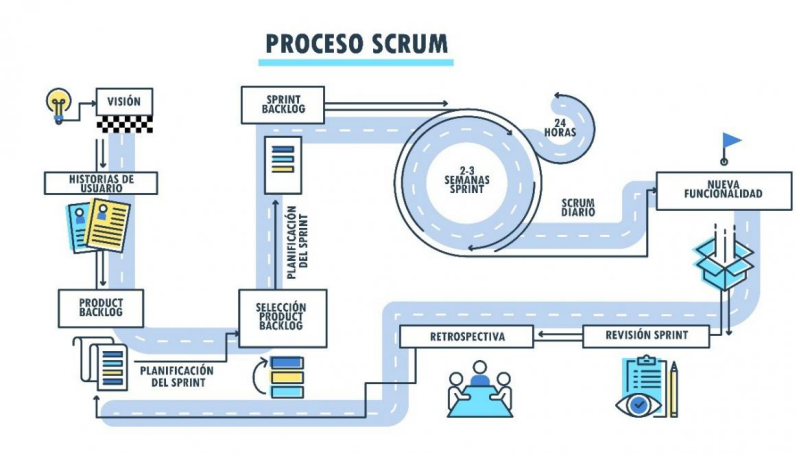


Figura 5.1: Diagrama do proceso de scrum

5.1.4 Adaptación de Scrum ao proxecto

Como se pode deducir, ao tratarse dun Traballo de Fin de Grao; Scrum non se pode aplicar de forma directa, pois como se vén indicando é unha metodoloxía pensada para un equipo multidisciplinar. Para solventar este problema farase unha adaptación de Scrum coas seguintes catacterísticas:

- Os *Sprints* terán unha duración de dúas semanas, co obxectivo de ter resultados usables con frecuencia.
- O sistemas de reunións vese completamente cambiado, pois non ten sentido realizar reunións diarias nun traballo individual. No seu lugar farase unha autoanalise todos os días e reunións periódicas cos directores do TFG.
- Dado que todo o traballo recae sobre a mesma persoa, esta representará todos os roles. De forma complementaria os directores do TFG poderán actuar como supervisores externos ou como cliente cando sexa preciso.

Capítulo 6

Planificación e estimación de custos

Unha vez xa escollida a metodoloxía e despois de saber como será o ciclo de vida do proxecto, podemos comezar a planificar como separaremos as diversas fases do proxecto, o tempo dedicado a cada incremento, os recursos de cada incremento, etc. Ademais, asociada a esta planificación podemos realizar a estimación dos custos que se terán na realización deste sistema, pois xa coñecemos os recursos e ferramentas que precisaremos, así como o tempo de emprego que faremos deles.

6.1 Planificación

Para a realización deste traballo tomouse como base un horario estándar de 20 horas semanais repartidas en 5 xornadas de 4 horas. Ademais, para simplificar a organización, como se trata dun traballo feito por un alumno, a organización das tarefas só ten en conta as fins de semana, pero non os festivos.

Cabe destacar que no *Sprint* 0, centraremonos na análise de requisitos inicial, así como a busca das ferramentas e tecnoloxías que usaremos para o desenvolvemento do sistema. Debido a esto non contaremos con ningunha saída por parte deste incremento.

Ademais deste incremento inicial vanse realizar outros 5 *Sprints*. O primeiro centrarase na creación do analizador léxico-sintáctico. O segundo e terceiro encargaranse da creación do servizo que o cliente terá que instalar no seu servidor. E por último, o cuarto e o quinto consistirán na creación da visualización que terá o usuario final do sistema á hora de ver o taboleiro de información no seu dispositivo.

Na figura 6.1 móstrase a imaxe do diagrama de Gantt que inclúe os *Sprints* xunto cos obxectivos de cada un, os roles que participan en cada actividade, as datas estimadas e o custo esperados.

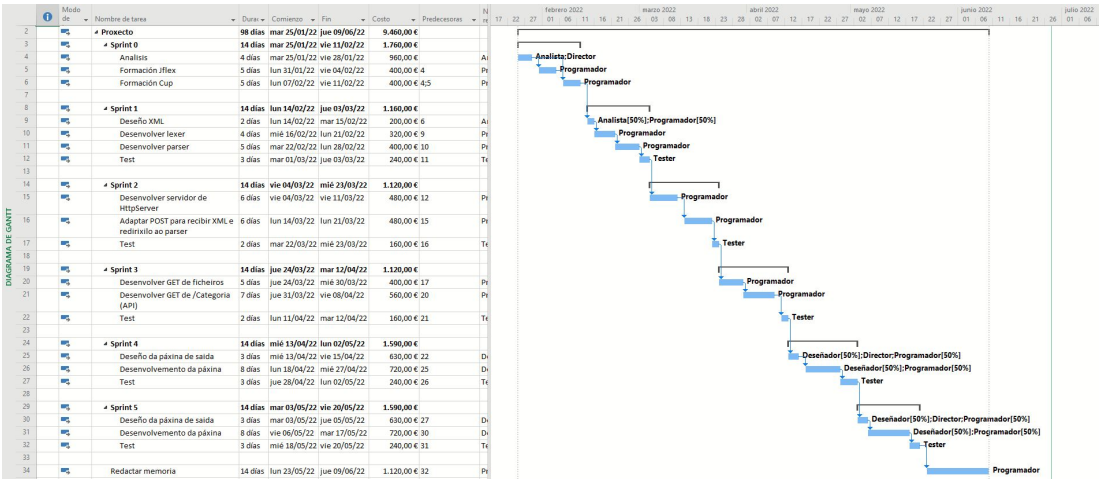


Figura 6.1: Diagrama de Gantt

Tal e como se pode observar no susodito diagrama de Gantt da figura 6.1 as datas de comezo e fin propostas son o 25/1 e o 9/6. Estas datas foron escollidas tendo en conta que o alumno deberá acompasar a realización do proxecto coas clases, polo que se buscou que fose posible realizar o traballo sen ser un tempo excesivamente axustado. Tamén se pode observar a existencia 5 roles diferentes:

- **Director**: é o perfil que representa aos directeres deste TFG.
- **Analista, Diseñador, Programador e Tester**: son os diversos perfís que representarán ao longo do proxecto ao alumno.

Cabe destacar que, con respecto ás datas propostas, os *Sprints* 0 e 1 se realizaron nun tempo algo menor que o estimado, pero por mor dos exames e entregas de prácticas finais do alumno esta tempo sobrante usouse para a realización do último *Sprint* e a realización desta memoria, polo que a data de finalización do proxecto acabouse retrasando un par de días.

6.2 Estimación de custos

Á hora de facer o cálculo dos custos estimados do proxecto deberemos separalos en custos por recursos materiais e humanos. Os primeiros son todos aqueles recursos que se deban comprar para o proxecto unha única vez, sen variar o seu valor dependendo das horas de uso. Os custos humanos son os salaríos que se lle deben pagar a todos os traballadores segundo o tempo dedicado ao proxecto.

6.2.1 Recursos materiais

No tocante aos materiais todo o *software* empregado é gratuito, tal e como se describe na sección de ferramentas empregadas (Cap. 4), polo que o único material a ter en conta sería o equipo utilizado para o desenvolvemento e probas do sistema. Neste caso usouse un único equipo como servidor e cliente tanto para desenvolvemento como para probas. Así pois a estimación de custos materias do proxecto é a indicada na táboa 6.1 .

Recurso	Custo
Equipo	900€
Software	0€

Táboa 6.1: Custos dos recursos materiais

6.2.2 Recursos humanos

Para a valoración dos custos dos traballadores estimouse o seu salario respectando os mínimos existentes segundo o BOE [31] do *XIX Convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos.* Con estes datos pódense facer os cálculos presentes na táboa 6.2, obtendo un custo total de 9.460€.

Recurso	Custo/hora	Horas	Custo total
Director	30€/H	40	1200€
Analista	30€/H	20	600€
Deseñador	25€/H	44	1100€
Programador	20€/H	276	5520€
Tester	20€/H	52	1040€

Táboa 6.2: Custos dos recursos humanos

Capítulo 7

Desenvolvemento

DURANTE este capítulo explicaremos os 6 incrementos levados a cabo neste proxecto. Para cada un deles comezaremos escollendo os obxectivos a cumprir, seguido de como se realizaron e das probas que se fixeron para comprobar o seu correcto funcionamento.

7.1 Incremento 0

Este incremento ten unha estrutura lixeiramente diferente ao resto, pois trátase do incremento inicial. Ao longo deste *Sprint* analizaremos o problema en profundidade para determinar os requisitos, seguido dun estudo das ferramentas que se deberán usar para o desenvolvemento da solución.

7.1.1 Análise

O proxecto deu comezo mediante unha reunión cos directores do TFG. Nesta reunión falouse das diversas funcionalidades que debería ter o sistema para que cumpra as nosas expectativas. A decisión foi crear un sitema que consista nunha API que permita recibir, almacenar e categorizar datos xunto información sobre o seu deseño na vista, así como envialos de forma ordenada. O sistema tamén funcionará como servidor, permitindo acceso a diversos recursos. Debese incluir unha forma de visualizar a información da almacenada na API para permitir a visualización por parte do usuario final. Finalmente, para suplir as necesidades esperadas polo cliente tamén se deben ter en conta os seguintes criterios:

- O usuario final que visualice a información debe poder acceder de forma sinxela, sendo beneficioso que o poida facer dende un navegador, sen necesidade de ningún software adicional, permitindo así ser usado dende calquera tipo de dispositivo con acceso á rede, como por exemplo paneis informativos.

- Buscase que o formato dos datos a subir sexa sinxelo, así calquera usuario autorizado poderá subir as novas noticias ao servicio sen necesidade de ningún especialista.
- O sistema debe poder soportar varias visualizacións diferentes, por exemplo, un mesmo servizo debe poder facer ao mesmo tempo un resumo xeral de noticias, un de deportes e un de economía.

Dadas estas especificacións decidíronse tomar as seguintes decisión sobre o proxecto:

- A entrada do sistema será un XML, pois trátase de un formato altamente sinxelo que pode ser enviado por calquera persoa sen ningún coñecemento da materia dende calquera programa que permita facer POST.
- Para permitir o seu acceso dende o navegador á visualización por parte do cliente final, a parte visual será realizada en HTML, CSS e JavaScript, aínda que se deixa aberto ao uso de outras tecnoloxías se o cliente o prefire.
- A parte visual será mediante páxinas modelo, ás cales o usuario accederá a través da función servidor e que accederán á API usando JavaScript.
- A relación entre as páxinas modelo e a API o formato que se usará para a información nese proceso será JSON.
- A extracción de datos do XML farase mediante un analizador léxico-sintáctico.

Tendo en conta estas decisións o fluxo de datos do sistema queda tal e como se mostra na figura 7.1, onde se pode observar:

- **Servizo**: *main* do programa que se encarga de enviar os recursos solicitados ou redirixir as peticións da API ao xestor de noticias cando sexa preciso.
- ***Parser***: compoñente encargada de realizar a análise léxico-sintática do XML antes de enviar a información nova ao xestor de noticias.
- **Xestor de noticias**: módulo do que depende a organización e almacenaxe de toda a información recibida polo sistema.
- **Recursos**: encargado do almacenamento de todos os recursos utilizados polas noticias. O servizo só pode ler deste compoñente do sistema.

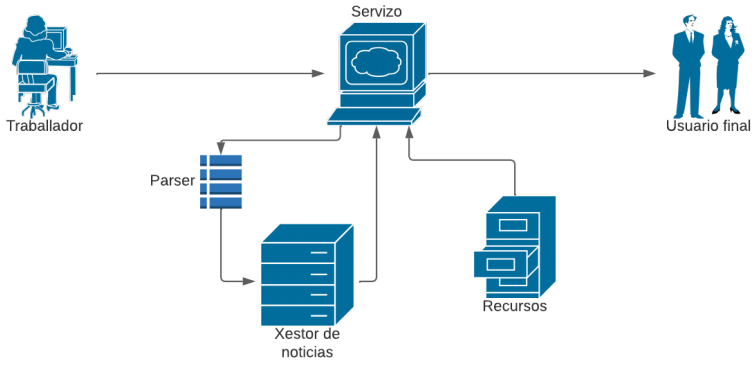


Figura 7.1: Fluxo de datos do sistema

Unha vez xa temos os requerimentos do sistema e un pequeno bosquexo de como vai ser a súa estrutura e funcionamento debemos escoller as ferramenta que utilizaremos para o seu desenvolvemento. Por unha banda empregamos tecnoloxías como XML, JSON, HTML, CSS e JavaScript; os cales escollemos para cumplir coas necesidades do sistema. Agora falta definir a linguaxe que conformará o código principal do servizo e o xestor de noticias, así como as ferramentas para crear o *parser*.

Como se trata dun servizo que fai uso do protocolo HTTP decidiuse usar Java como linguaxe de programación no que se desenvolva este proxecto. Tomouse esta decisión pola facilidade de creación deste tipo de servizos nesta plataforma grazas á súa facilidade de realizar conexión mediante HTTP e á súa inxente cantidade de bibliotecas, que facilitan realizar este traballo. Ademais tamén foi determinante para a súa elección o feito de ter un coñecemento previo desta linguaxe.

No tocante ao analizador temos que escoller unhas ferramentas que se adapten á linguaxe escollida, Java. Para a parte léxica escolleuse JFlex por ser unha adaptación de Flex, ferramenta empregada na materia de PL da que xa se teñen coñecementos previos. Pola outra banda, tal e como se indica na sección 4.2, débese escoller un analizador sintáctico compatible co noso analizador léxico. De entre os dous recomendados polos creadores de JFlex nós escollemos CUP, do cal non se ten coñecemento, pero pódese usar parte da experiencia conseguida de usar Bison [32], tamén durante a materia de PL.

Agora que xa temos escollidas as ferramentas a empregar só queda escoller un IDE no que realizar o proxecto e un sistema de versión no que gardalo. Para o primeiro decídese usar Eclipse, un contorno de desenvolvemento que soporta Java e C no que xa temos experiencia polo seu uso en varias materias da carreira. Para o sistema de versións usaremos GitHub, unha plataforma gratuíta propiedade de Microsoft.

Unha vez escollidas todas as ferramentas e tecnoloxías a empregar durante o proxecto só nos queda realizar a instalación de Eclipse, JFlex e Cup. Adicionalmente tamén instalaremos

Postman, unha aplicación que nos permite realizar diversos tipos de peticións HTTP personalizando os seus valores, o cal será de gran axuda para a realización das probas do sistema.

Finalmente, a pesar de que xa se ten coñecemento de Flex e Bison útiles para JFlex e CUP, decidiuse empregar parte do tempo desta primeira iteración en formarse sobre estas dúas tecnoloxías, asegurándose un mínimo de calidade e soltura á hora de realizar o *parser* do sistema.

7.2 Incremento 1

Unha vez rematado o incremeto inicial pódese comezar co *Sprint* 1, no cal se comezará co desenvolvemento do proxecto.

7.2.1 Seguemento e análise

Tras haber finalizado o primeiro incremento e xa ter escollidas as ferramentas a empregar débese analizar a situación para planificar os obxectivos principais deste *Sprint*.

Neste caso chegouse á conclusión que o máis lóxico e útil sería comezar creando a clase básica que conteña a información das novas que o sistema recibe, así como o seu método de creación, o *parser*.

Para a realización desta parte necesitouse analizar as necesidades de información que requería o noso contexto. Tras esta análise chegouse á conclusión de que cada fragmento de información se podía partir nos seguintes campos:

- **Título**: título da noticia en cuestión.
- **Autor**: o autor da información.
- **Texto**: o contido da noticia.
- **Data e hora**: indican o momento no que se redactou a noticia. Úsase para ordenalos de forma cronolóxica, polo que é o único campo obrigatorio.
- **Imaxe**: unha imaxe asociada á noticia.
- **Vídeo**: un vídeo asociada á noticia.
- **Categoría**: categoría á que pertence a noticia.

O noso caso tratase de un contexto simple que require unha información reducida, se se desexa podense incluir multitude de campos adicionais: idioma, diversos campos de texto para multiple idiamos, enlaces, identificadores de noticia, etc.

Así mesmo tamén se escolleron 5 campos adicionais para personalizar a posta en pantalla de cada nova:

- **Tipo**: etiqueta que pode coller calquera valor que pode servir para identificar certo tipo de noticias. Algúns exemplos poden ser: importante, destacable, noticia interna,etc Pode funcionar a modo de segunda categoría.
- **Cor**: cor de fondo que terá a noticia.
- **CorLetra**: cor que terá a letra.
- **Letra**: a fonte coa que se escribirá a noticia.
- **Posición**: posición da noticia dentro da páxina. Útil naqueles casos nos que unha mesma categoría pode aparecer en dous sitios diferentes. Por exemplo, se dentro dunha paxina de noticias temos unha segunda páxina unicamente de deportes, este campo podería indicarnos en que zona desta páxina se debe crear.

Ao igual que no caso dos campos de información, os campos de deseño tamén se poden ampliar de infinidade de formas, podendo engadir tamaño, transición, animación, iconos,etc

Todos estes campos son optativos, a excepción da data que sempre será necesaria para poder ordenalos cronoloxicamente.

Unha vez escollidos os obxectivos desta primeira iteración de desenvolvemento comezaremos co deseño necesario.

7.2.2 Deseño

A primeiro paso é deseñar a clase que albergará toda esta información de tal forma que se poida adaptar se o cliente o necesita. Para iso creamos unha interface que todas as clases de información deben implementer. Esta interface obriga a implementar as seguintes funcións:

- ”public int compareTo” da interfaz Comparable<Object> de tal forma que ordene a información de maior a menor prioridade.
- ”public String getCategoria”, que devolve a categoría da información.
- ”public String toJson”, que devolve o String do JSON que representa esa información.
- ”public boolean hasError”, que devolve ”true” se houbo algún erro recoñecible durante o *parse*.
- ”public String error”, que devolvese o erro recoñecible durante o *parse*.

Unha vez deseñada esta interface só quedaría desenvolver a clase *MyInformación* cumprindo as características recollidas durante a análise.

Como xa se decidiu no incremento 0 (sección 7.1) os datos de entrada serán representados mediante XML, polo que se debe deseñar como será para o noso caso concreto. Finalmente decidiuse que para simplificar a gramática, e polo tanto a súa creación e análise, o noso XML ten que cumprir as seguintes normas:

- O XML non pode conter comentarios.
- O campo raíz pode ter calquera nome.
- Os campos que se terán en conta denominanse: ”Autor”, ”Titulo”, ”Texto”, ”Imagen”, ”Video”, ”Categoria”, ”Fecha”, ”Tipo”, ”Color”, ”Letra”, ”ColorLetra” e ”Posicion”; calquera outro campo será ignorado.

```
<info>
    <Autor>Brais</Autor>
    <Fecha>25-06-2022 19:20:30</Fecha>
    <Color>yellow</Color>
</Info>
```

Listing 7.1: Exemplo correcto

- Todos os campos son optativos agás a raíz e o campo ”Fecha”.

```
<info>
    <Autor>Brais</Autor>
    <Color>yellow</Color>
</Info>

**************************************************************************
Erro, falta o campo "Fecha".
```

Listing 7.2: Exemplo de erro por falta de fecha

- Os campos poderán estar baleiros agás a fecha, que sempre debe seguir o formato ”dd-MM-yyyy hh:mm:ss”.

```
<info>
    <Autor></Autor>
    <Fecha>25-06-2022 19:20:30</Fecha>
    <Color></Color>
</Info>
```

Listing 7.3: Exemplo correcto con un campo nulo

- Os campos non poden levar atributos.

```
<info>
    <Autor nome="Brais"></Autor>
    <Fecha>25-06-2022 19:20:30</Fecha>
    <Color>yellow</Color>
</Info>

**********************************************************************
Erro, o campo "Autor" non pode levar atributos.
```

Listing 7.4: Exemplo de erro por ter un campo con atributo

- Ningún campo excepto o raíz pode ter campos dentro.

```
<info>
    <Autor>
        <Nome>Brais</Nome>
    </Autor>
    <Fecha>25-06-2022 19:20:30</Fecha>
    <Color>yellow</Color>
</Info>

**********************************************************************
Erro, o campo "Autor" non pode ter campos anidados.
```

Listing 7.5: Exemplo de erro por ter campos anidados

Ademais de todas estas normas tamén debe cumprir coas normas propias de XML, así como incluir a cabeceira para saber codificación e idioma empregados.

7.2.3 Desenvolvemento

Unha vez que xa temos deseñada a interface ”Informacion” e a normas dos nosos XMLs podemos comezar a desenvolver o código correspondente deste primeiro incremento.

Comezamos creado a clase ”MyInformacion”, a cal implemente a interface ”Informacion”. Nela podemos atopar un atributo para cada un dos 11 campos diferentes que podemos recibir; un *boolean* ”error”, que indica se houbo erros durante o *parser*, e un *string* ”errorMessage”, coa información do erro incluído. Ademais tamén atoparemos a implementación de cada un dos métodos da interface e dous construtores, un para a creación con datos correctos e outro para a creación cando hai un erro. Ademais, no caso do primeiro construtor pódese dar o caso de que a data e hora non teñan o formato adecuado, xerando un erro que funciona da mesma forma que se se tratase dun erro por parte do *parser*.

No caso da función ”compareTo” chámase a función co mesmo nome da clase ”DateTime”, de tal forma que ordene as noticias de máis recentes a máis antigas.

Para a función ”toJson” cabe destacar o uso da biblioteca ”commons-text”, explicada dentro da sección 4.1, que nos permite formatear o *string* de tal forma que todos os seus caracteres sexan válidos para a notación de JSON e HTML.

Tras rematar a clase ”MyInformacion” quédanos crear o analizador léxico-sintáctico. O primeiro paso é realizar o analizador léxico con JFlex (sección 4.2). Crearemos un *lexer* capaz de recoñecer as cadeas coa prioridade indicada na táboa A.1.

Lexema	***Token* xerado**	**Aclaracións**
</X>	FINCAMPO	*X* é unha cadea de texto que comeza por unha letra e que non contén ningún destes símbolos: <, \n, \r, \t, ”, / ou espazo.
<Autor>	AUTOR	
<Titulo>	TITULO	
<Texto>	TEXTO	
<Imagen>	IMAGEN	
<Video>	VIDEO	
<Categoria>	CATEGORIA	
<Fecha>	FECHA	
<Tipo>	TIPO	
<Color>	COLOR	
<Letra>	LETRA	
<ColorLetra>	COLORLETRA	
<Posicion>	POSICION	
<X>	CAMPO	*X* é unha cadea de texto que comeza por unha letra e que non contén ningún destes símbolos: <, \n, \r, \t, ”, / ou espazo.
BLANCOS		unha cadea de texto formada únicamente por estes símbolos: \n, \r, \t ou espacio.
TEXTOBASICO	TEXTOBASICO	unha cadea de texto formada por símbolos diferentes de ”<”.

Táboa 7.1: Lexemas recoñecidos polo lexer ordenados por preferencia

Unha vez xa temos realizado o *lexer* o seguinte paso é crear o *parser* que usará os *tokens* que definimos e comentamos no bloque anterior. Para iso creamos un conxunto de regras sintácticas , tal e como se mostra na listing 7.6, para as cales o símbolo inicial é ”ini”.

```
ini::= CAMPO body;

body ::=
   linea body
 | linea FINCAMPO
;

linea  ::= info FINCAMPO;

info ::=
    AUTOR TEXTOBASICO
  | TITULO TEXTOBASICO
  | TEXTO TEXTOBASICO
  | IMAGEN TEXTOBASICO
  | VIDEO TEXTOBASICO
  | CATEGORIA TEXTOBASICO
  | FECHA TEXTOBASICO
  | TIPO TEXTOBASICO
  | COLOR TEXTOBASICO
  | LETRA TEXTOBASICO
  | COLORLETRA TEXTOBASICO
  | POSICION TEXTOBASICO
  | CAMPO TEXTOBASICO
  | AUTOR
  | TITULO
  | TEXTO
  | IMAGEN
  | VIDEO
  | CATEGORIA
  | FECHA
  | TIPO
  | COLOR
  | LETRA
  | COLORLETRA
  | POSICION
  | CAMPO
  ;
```

Listing 7.6: Regras da grámatica en BNF

Dentro do parser, a medida que se van cumprindo as regras da gramática, vanse almacenando os datos que se analizan cada vez que se entran nas regras de *info* ata que finalmente

se chama ao construtor da clase ”MyInformación” cos datos adquiridos ou co construtor de erro en caso de non existir campo ”Fecha” ou que nun campo non coincidan cabeceira inicial e final.

Unha vez finalizado o *parser* xa teriamos un sistema funcional capaz de transformar XML entrantes en instancias de clase de Java que posteriormente se poden transformar en JSON ou detectar erros en caso de que estean mal formados.

7.2.4 Probas

Durante as probas desta iteración hai que comprobar o correcto funcionamento do analizador. Para levar a cabo esta acción debemos centrarnos en dúas cousas a comprobar: os datos plásmanse correctamente na instancia da clase ”MyInformacion” e os erros detéctanse correctamente seguindo as regras impostas durante a fase de deseño.

No primeiro caso tomáronse unha serie de XML ben formados para ser analizados polo *parser* e posteriormente impresos por pantalla usando a función ”toJson”. Cos resultados dados pódese comprobar manualmente ou mediante conversores de JSON a XML que os datos de entrada e saída son os mesmos, tendo en conta a adapatación de caracteres para JSON e HTML.

Para a detección de erros simplemente hai que intentar analizar XMLs que non cumpran coas regras estipuladas, buscando que a función ”error” devolva o erro esperado, no caso dos que son recoñecidos, ou que salte unha excepción no resto.

7.3 Incremento 2

Agora que xa temos a nosa clase base que conforma os datos que o noso sistema usará e manexará podemos comezar co servizo coma tal.

7.3.1 Seguemento e análise

Unha vez completada a clase ”MyInformacion” xunto co analizador de XML xa temos unha forma de usar os datos de entrada dentro de Java. Tras analizar os obxectivos finais do proxecto decidiuse que neste incremento se crearía o servizo como tal e as funcionalidades encargadas da petición POST, de tal forma que o resultado deste *Sprint* debería ser un servizo capaz de recibir os datos do usuario.

Durante a análise chegouse á conclusión de que debiamos cumprir os seguintes obxectivos:

- O servizo debe cribar as IPs que fagan POST de tal forma que só poidan enviar datos aquelas que estean permitidas.

- O servizo debe poder configurar o porto, o directorio base, as categorías existentes e o número máximo de noticias por categoría sen necesidade de modificar o código.

7.3.2 Deseño

A nivel de deseño houbo que abordar dous obxectivos en especial: a estrutura do servizo e a independencia con respecto ao analizador e aos parámetros indicados na análise.

Para a estrutura do servizo decidiuse baseala na biblioteca HttpServer, explicada na sección 4.1, para a cal teremos que crear unha clase principal que conteña o *main* e na cal se crea o servizo e outra clase que implemente a interface "HttpHandler".

Para a independencia decidiuse crear unha clase chamada "MyProperties" que poida ler dun ficheiro os parámetros que pon na táboa 7.2 así como ser a encargada de crear a instancia do *parser*. No caso do campo "IPPOST" as IPs poderan ser IPv4 ou IPv6 e poderán estar definidas de dúas formas. A primeira e máis básica é poñer a IP específica lista, e esta será aceptada. A outra forma é escribir dúas IPs separadas por un "-", neste caso aceptaranse todas as IPs que sexan maiores ou iguais á primeira e menores ou iguais á segunda; no caso de que a segunda sexa menor que a primeira ou que haxa mais de dúas IPs separadas por guións, estas non aceptarán ningunha IP. Estes dous métodos pódense combinar dentro da mesma lista de IPs. Un exemplo de lista válida sería:

0:0:0:0:0:0:0:0-0:0:0:0:0:0:0:f;122.0.0.2-133.0.0.0;185.12.45.95;aa:10:51:45:b2:45:85:d5:5d

Con todo isto conseguiriamos unha estrutura como a que se mostra na figura 7.2.

Parametro	Utilidade
PORT	o porto desde o que se debe acceder ao servizo
DIR	o directorio base do servidor
CATEGORIAS	a lista de categorías separadas por ";"
CATEGORIASIZE	o número de noticias máximas por categoría
IPPOST	as IPs permitidas para POST separadas por ";"

Táboa 7.2: Parámetros lidos por ficheiro

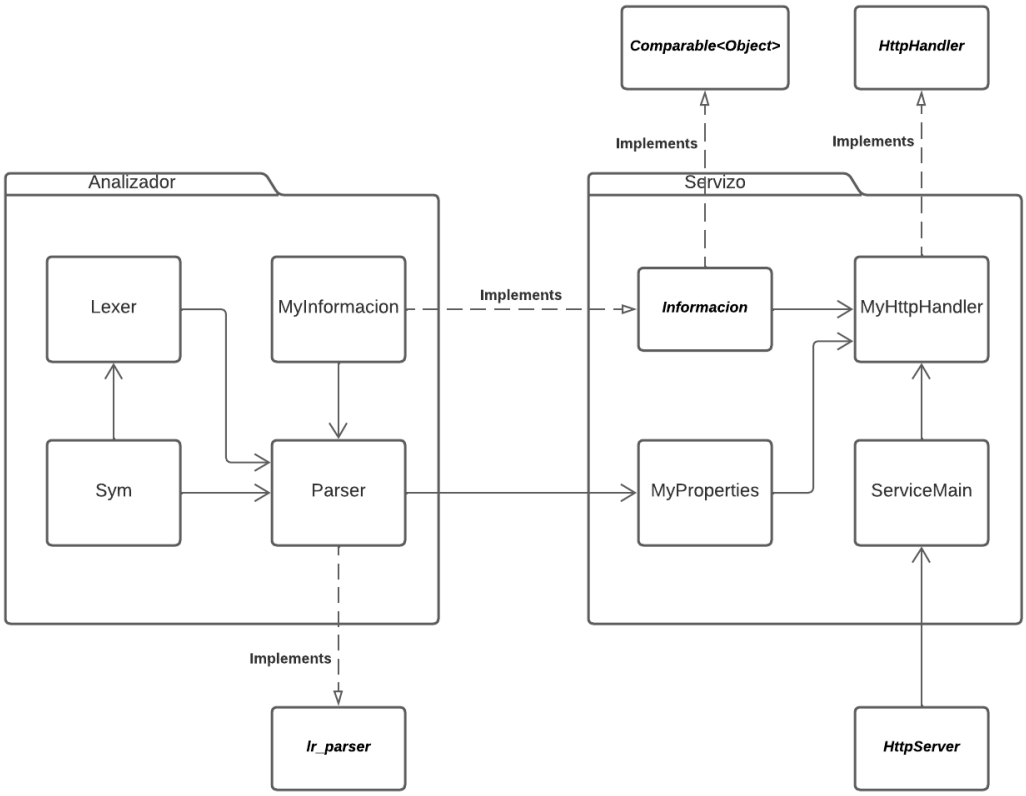


Figura 7.2: Estrutura do sistema no incremento 2

7.3.3 Desenvolvemento

Para comezar co desenvolvemento comezamos coa creacion da clase ”MyProperties”. Dentro desta clase creáronse dous métodos estáticos; ”getProperties()”, que devolve a instancia de clase ”Properties” que permite ler os campos do arquivo ”Properties.txt” que existe na raíz do proxecto, e ”getParser(Reader r)”, que devolve o *parser* encargado de analizar o contido de *Reader* ”r”.

O seguinte paso é crear ”MyHttpHandler”. Esta clase ao implementar a interface ”HttpHandler” debe ter o método ”handle”, que recibirá a información de cada petición que o servizo reciba e terá que analizala e respondela. Dentro deste método o que faremos será detectar o tipo de petición para de seguido redirixila a outra función que xestione ese tipo de peticións se están soportadas; neste caso a única opción é que ao recibir unha petición de POST esta redirixirase á función ”handlePostRequest”, en caso contrario devolverá erro 405. A función ”handlePostRequest” fará o seguinte:

1. Recupérase a IP do cliente que está intentando facer o POST.

2. Úsase ”MyProperties” para ler a lista de IPs permitidas e comprobará se a do cliente está entre elas:
    - Se a IP non se atopa na lista devólvese erro 403 e pecarase a conexión.
    - Se a IP si está na lista continúase coa execución.
3. Créase un *Reader* co corpo da petición HTTP.
4. Úsase ”MyProperties” para conseguir un *parser* que analice o *Reader*.
5. Úsase o analizador para analizar o XML, podendo darse 4 casos:
    - Devolve null, polo que houbo un erro descoñecido durante o *parser* e devólvese erro 400.
    - Devolve unha instancia de ”Informacion” cun erro coñecido, devólvese erro 400 acompañado dunha descrición do erro.
    - Devolve unha instancia de ”Informacion” correcta, devólvese código 201 *Created* e móstrase por pantalla o JSON resultante.
    - Lánzase unha excepción, polo que houbo un erro descoñecido durante o parser e devólvese erro 400.

Finalmente só queda a creación do *main* principal no ”ServiceMain”. Este *main* encargarase de coller o porto polo cal se vai acceder ao servizo mediante a clase ”MyProperties” para crear o ”HttpServer” que ofrecerá o servizo usando o ”MyHttpHandler” que se acaba de explicar.

Implementadas estas tres clases xa temos creado un sistema capaz de recibir XML mediante peticións POST de HTTP para analizalas e posteriormente mostrar a súa tradución a JSON. Como resultado deste desenvolvemento temos un sistema cuxo diagrama de clase podemos ver na figura 7.3.

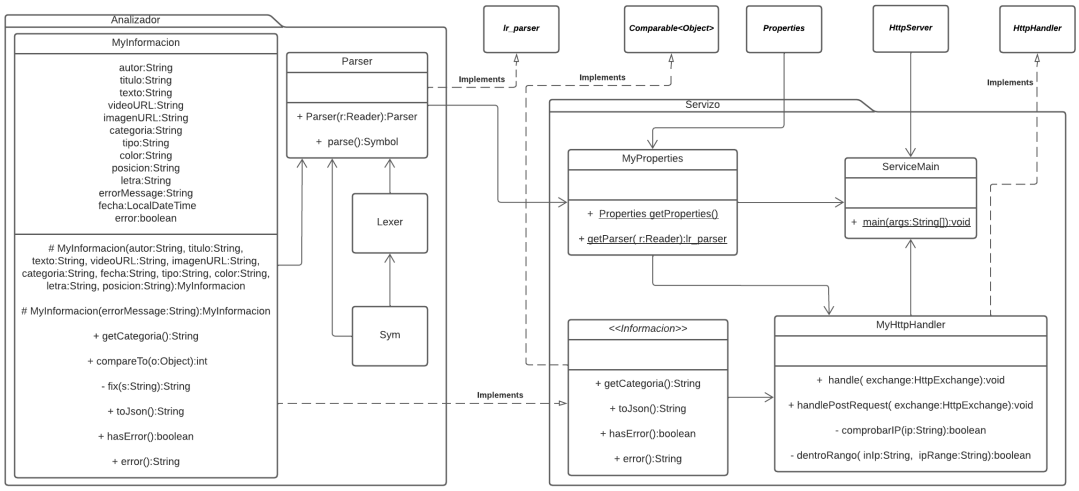
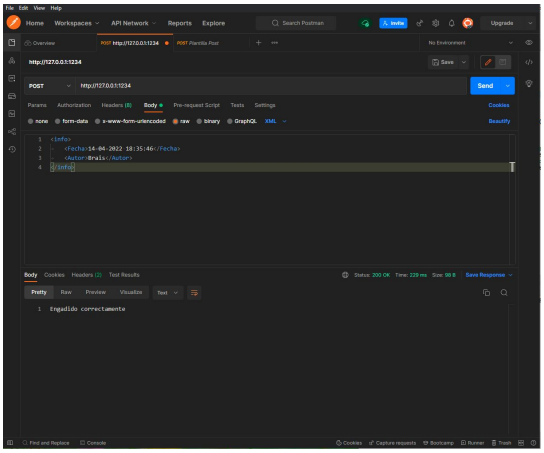


Figura 7.3: Diagrama de clases do incremento 2

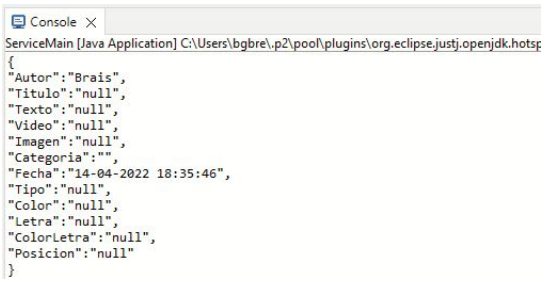
7.3.4 Probas

As probas neste caso son moi similares ás da iteración 1, centrándose en que os datos se reciben, analicen e extraian de forma correcta e que os erros se produzan da forma esperada.

Neste caso, para probar todo mediante HTTP usouse o programa Postman, que permite facer todo tipo de peticións personalizando os seus parámetros e visualizando a súa resposta. Podemos ver algúns exemplos nas figuras 7.4 e 7.5.

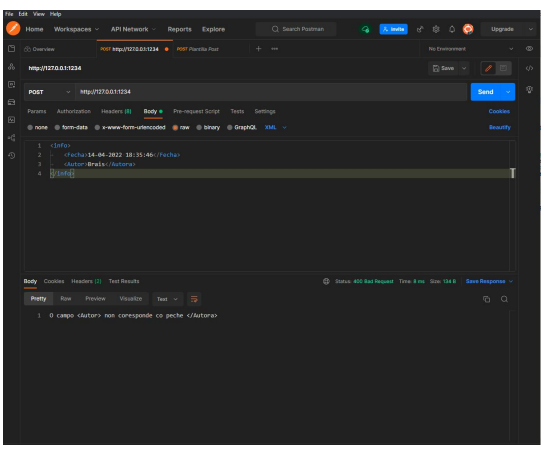


(a) Pantalla de Postman

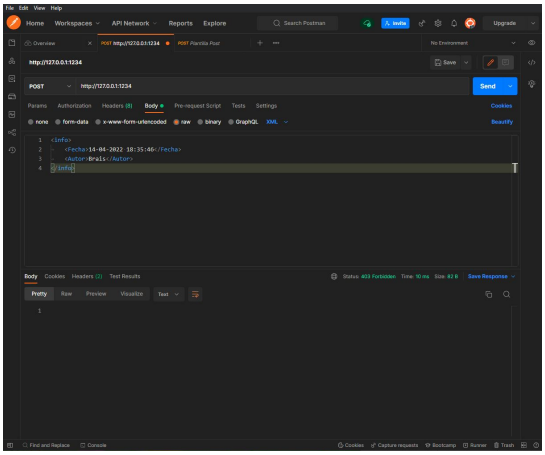


(b) JSON producido pola petición

Figura 7.4: Petición de POST correcta



(a) Erro no peche de un campo



(b) IP non permitida para realizar POST

Figura 7.5: Peticións de POST incorrectas

7.4 Incremento 3

Unha vez creado e probado o incremento 2 xa dispoñemos dun servizo capaz de recibir peticións POST, polo que neste incremento crearemos da resposta á peticións GET, finalizando así as funcionalidades que o recurso debe ter.

7.4.1 Analise e seguemento

Tendo xa lista a funcionalidade que permite a recepción de información debemos crear a parte que se encargará de almacenar, organizar e entregar os datos. Para esto debemos crear no servizo unha funcionalidade que permita a recepción de peticións GET.

Dentro deste incremento podemos atopar dúas seccións, unha funciona como unha API, que denominaremos "/Categoria" por ser así o seu enlace, e autra que funciona como un server de recursos. Así mesmo deducironse dous punto importantes a ter en conta:

- A API debe proporcionar unha forma de obter as categorías existentes.
- O servidor debe soportar como mínimo o envío de HTML, CSS, JavaScript, imaxe e vídeo.

7.4.2 Deseño

Nun primeiro lugar comezamos solucionando o problema de que queremos que o sistema teña 3 posibles tipos de petición GET: solicitar un recurso, solicitar as noticias dunha categoría ou solicitar a lista de categorías. Para solucionar este problema decidiuse separar a xerarquía nas tres posibilidades que se poden ver na táboa 7.3.

Xerarquía	Función
/Categoria ou */Categoria/*	Devolve a lista de categorias existentes. (Non ten en conta maiúsculas ou minúsculas)
/Categoria/X	Devolve a lista de noticias da categoria *X* se existe, se non devolve erro 404. (Non ten en conta maiúsculas ou minúsculas)
/X	Devolve o recurso que se atopa na dirección *X* tomando como base o directorio que aparece no documento "Properties.txt". Só se usará este método se a petición non encaixa nun dos dous anteriores.

Táboa 7.3: Xerarquía da petición GET

Por outra banda temos que abordar como se xestionarán as noticias que queremos gardar e devolver, polo que se propuxo a creación dun "manager" que creará e xestionará un grupo de categorías predefinidas no documento "Properties.txt". Este "manager" ademais debe seguir o patrón de instancia única, pois interésanos que sempre sexa o mesmo o que responda as peticións.

Tendo en conta a creación desta nova funcionalidade do sistema, a estrutura deste quedaría tal e como se ve na figura 7.6.

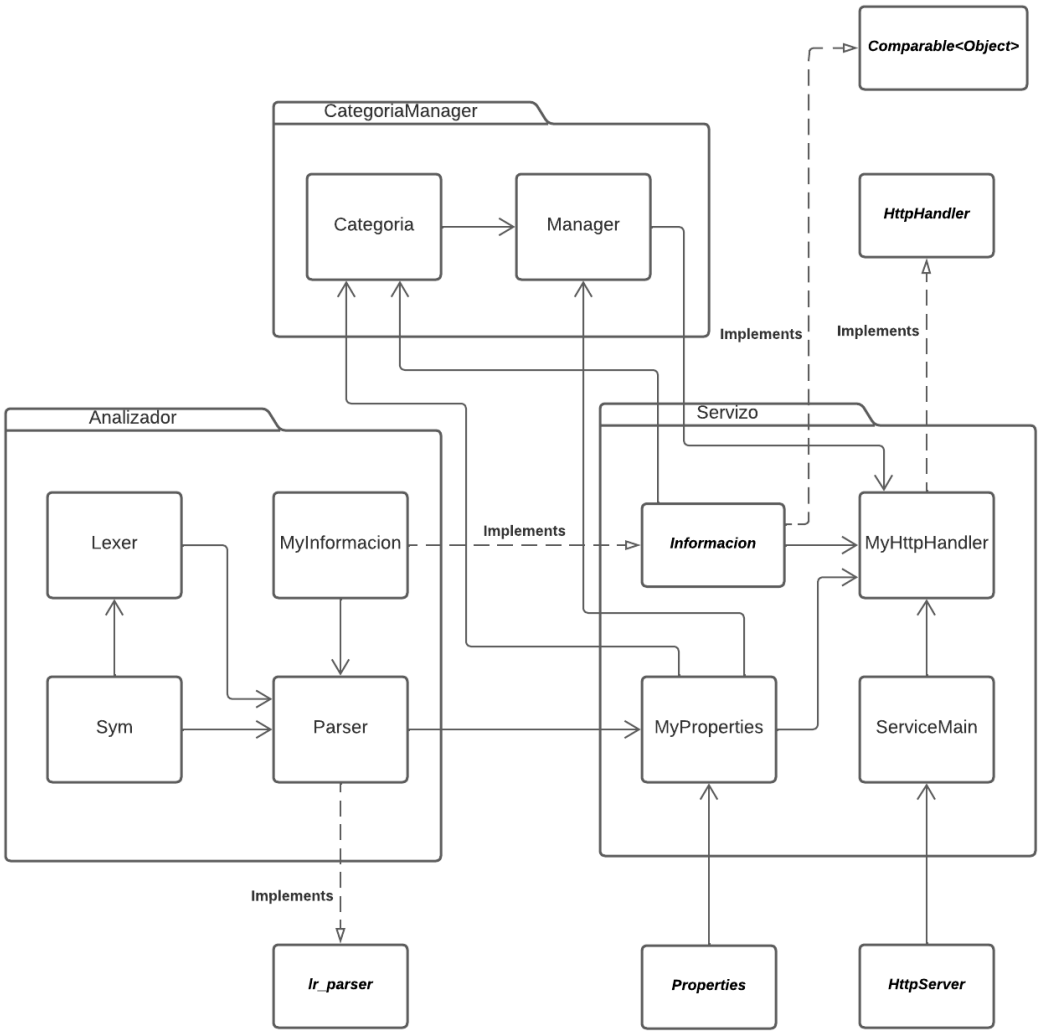


Figura 7.6: Estrutura do sistema no incremento 3

7.4.3 Desenvolvemento

Comezaremos o desenvolvemento coa creación da función ”handleGetRequest” da clase ”MyHttpHandler”. Esta función será chamada dende ”handle” cada vez que se reciba unha petición tipo GET. Neste primeiro instante comezaremos facendo que esta función sirva para mandar como resposta o recurso especificado sen ter en conta a ruta ”/Categoría”. A función seguirá os seguintes pasos:

1. Recuperar a ruta da petición HTTP.

2. Recuperar o directorio base mediante ”MyProperties”.

3. Localiza o ficheiro (directorio base + ruta) é comproba se existe, poden darse dúas opcións:
    - Se non existe, enviase erro 404 e acábase a resposta da petición GET.
    - Se existe, continúase coa execución.
4. Modifícanse os campos da cabeceira necesarios. Seguindo os obxectivos da fase de análise, é de especial interés modificar o campo ”Content-Type” para que corresponda co tipo do arquivo.
5. Asígnase o código 200 OK.
6. Asígnase o corpo da resposta co ficheiro.
7. Envíase a resposta da petición GET.

Para ampliar esta función para que teña en conta a sección de ”/Categoria” simplemente teremos que engadir un *if* despois do primeiro paso tendo en conta a ruta recuperada. A resolución desta petición significa facer os pasos do 4 ao 7 usando o resultado que o ”manager” devolva nas funcións que respondan ás necesidades recibidas. A única excepción será cando se solicite unha categoría inexistente, que terá que devolver erro 404.

Para finalizar o desenvolvemento deste incremento queda a creación do ”manager” e as categorías. Comezaremos con esta segunda.

A clase ”Categoría” é a encargada de almacenar e ordenar os datos que recibe sen realizar ningunha comprobación de se lle pertencen. Esta clase só consta dun atributo privado, que é a lista na que se garda a información, e dous métodos: ”add” e ”getList”. Como o seu nome indica, o primeiro serve para engadir unha instancia de ”Informacion” á lista; e o segundo devolve a lista. O método add non solo engade a noticia á lista, tamén se encarga de ordenala por prioridade (seguindo o criterio da implementación dada a ”Información”, no noso caso ”MyInformacion”) e eliminar as de menor importancia se hai máis noticias das permitidas segundo o campo ”CATEGRIASIZE” do ficheiro de propiedades.

A clase ”Manager” será a encargada de xestionar as diversas categorías, así como mandar a información á categoría que corresponda. Contará con 3 atributos: un ”Manager” estático, necesario para a implementación do patrón instancia única; un mapeado entre *string* e ”Categoria”, que nos permitirá relacionar cada categoría co seu nome, e un *array* de *strings*, que contén a lista de categorías existentes no ficheiro de propiedades no momento de creación do manager. Ademais conta coas seguintes funcións:

- ”Manager”: O construtor da clase manager. Encárgase de crear o mapeado das categorías da lista do documento de propiedades e tamén unha extra que recibirá o nome de ”Outras”. Este método é privado para implementar o patrón instancia única.

- ”getManager”: Método estático principal do patrón instancia única, se é a primeira vez que se chama crea un novo ”Manager”; se non, devolve o que creou anteriormente.
- ”add”: Engade a información recibida na categoría que devolve o método ”getCategoria” da clase ”Informacion”, se esa categoría non existe introdúcese na categoría ”Outras”.
- ”get”: Devolve o JSON que contén todas as noticias dunha categoría de forma ordenada.
- ”getCategorias”: Devolve o JSON que contén todas as categorías, incluíndo a categoría ”Outras”.

Finalmente falta engadir que tras finalizar o *parser* dentro de ”handleGetRequest” na clase ”MyHttpHandler” debemos redirixir a saída deste á función ”add” do Manager.

Engadindo estas dúas novas clases, as súas conexións e o novo método ”handleGetRequest” da clase ”MyHttpHandler” o novo diagrama de clases é tal e como se pode ver na figura 7.7.

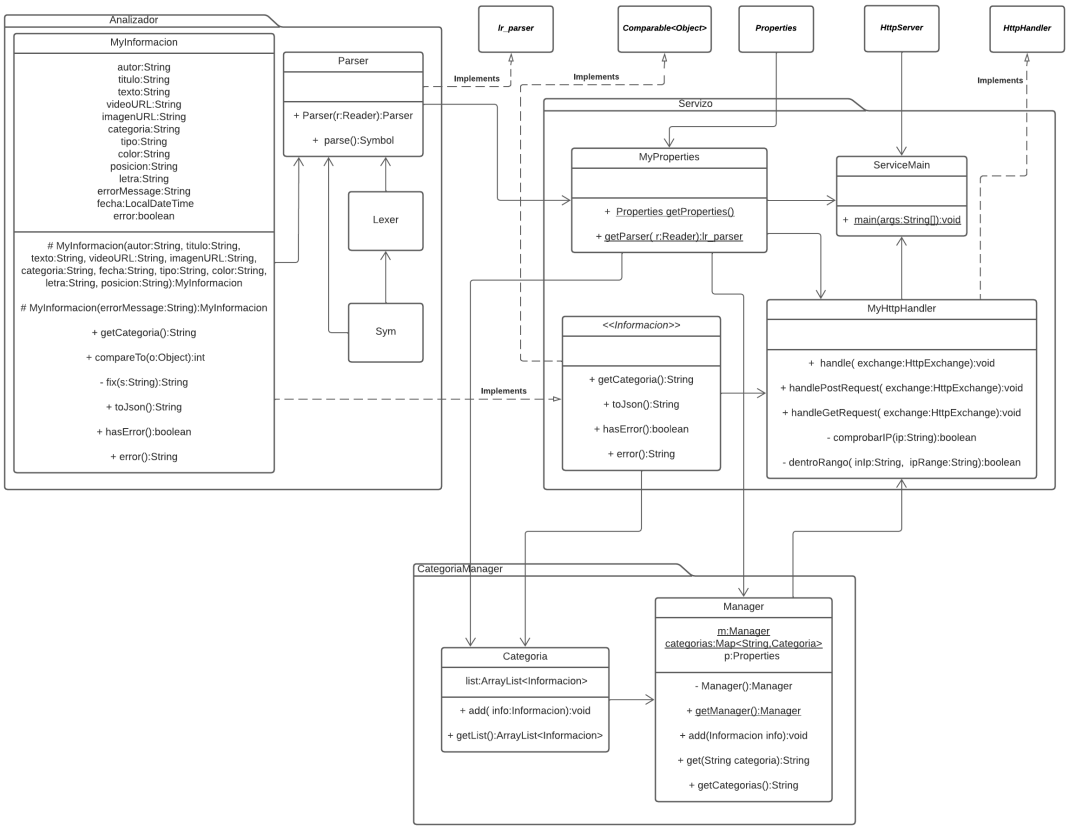


Figura 7.7: Diagrama de clases do incremento 3

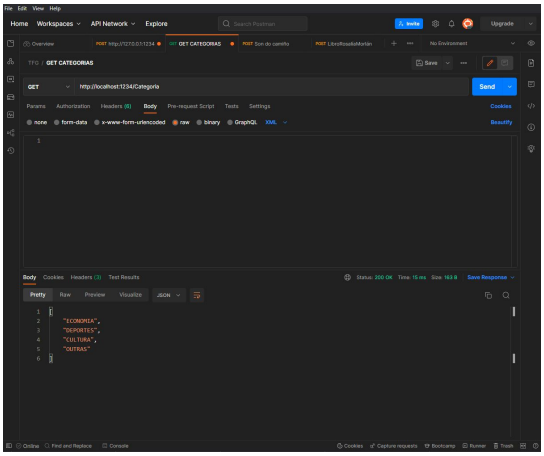
7.4.4 Probas

As probas desta parte do incremento son moi similares ás do anterior incremento. Centrarémonos en comprobar que cada unha das tres opcións de funcións que poden pasar con respecto á ruta funcionan correctamente. Para os exemplos da figura 7.8 e 7.9 usamos os seguintes XMLs:

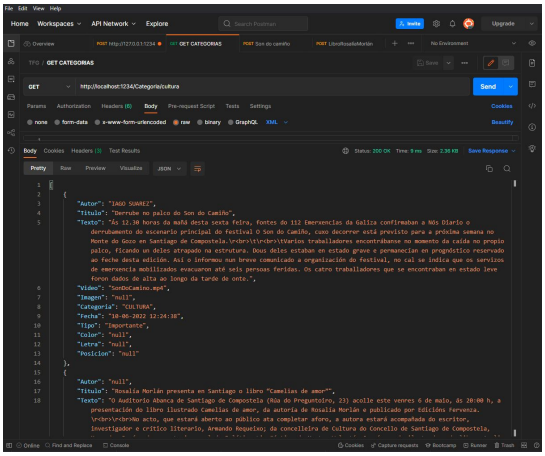
```
<Informacion>
  <Fecha>12-05-2022 11:08:00</Fecha>
  <Autor></Autor>
  <Categoria>Cultura</Categoria>
  <Titulo>Rosalía Morlán presenta en Santiago o libro “Camelias de
    ”amor</Titulo>
  <Texto>O Auditorio Abanca de Santiago de Compostela (Rúa do
    Preguntoiro, 23) acolle este venres 6 de maio, ás 20:00 h, a
    presentación do libro ilustrado Camelias de amor, da autoría de
    Rosalía Morlán e publicado por Edicións Fervenza.

No acto, que estará aberto ao público ata completar aforo, a autora
    estará acompañada do escritor, investigador e crítico literario,
    Armando Requeixo; da concelleira de Cultura do Concello de
    Santiago de Compostela, Mercedes Rosón; do secretario xeral de
    Política Lingüística da Xunta, Valentín García; e da ilustradora
    do libro, Amalia López Brea “”Luchi.

Contarase ademais co acompañamento musical de Manoele de Felisa e o
    Coro de Habaneras Rocha Forte. </Texto>
  <Imagen>https://www.noticieirogalego.com/wp-content/uploads/2022/
    05/cartel-camelias-estrada-555x391.jpg</Imagen>
  <Video></Video>
  <Tipo></Tipo>
  <Color>IndianRed</Color>
  <Letra>Lucida Handwriting</Letra>
  <ColorLetra>Ivory</ColorLetra>
  <Posicion></Posicion>
</Informacion>
```

```
<Informacion>
  <Fecha>10-06-2022 12:24:38</Fecha>
  <Autor>IAGO SUAREZ</Autor>
  <Categoria>Cultura</Categoria>
  <Titulo>Derrube no palco do Son do Camiño</Titulo>
  <Texto>Ás 12.30 horas da mañá desta sexta feira, fontes do 112
    Emerxencias da Galiza confirmaban a Nós Diario o derrubamento do
    escenario principal do festival O Son do Camiño, cuxo decorrer
    está previsto para a próxima semana no Monte do Gozo en Santiago
```

```
    de Compostela.

    Varios traballadores encontrábanse no momento da caída no propio
      palco, ficando un deles atrapado na estrutura. Dous deles
      estaban en estado grave e permanecían en prognóstico reservado
      ao feche desta edición. Así o informou nun breve comunicado a
      organización do festival, no cal se indica que os servizos de
      emerxencia mobilizados evacuaron até seis persoas feridas. Os
      catro traballadores que se encontraban en estado leve foron
      dados de alta ao longo da tarde de onte.</Texto>
    <Imagen></Imagen>
    <Video>SonDoCamino.mp4</Video>
    <Tipo>Importante</Tipo>
    <Color></Color>
    <Letra></Letra>
    <ColorLetra></ColorLetra>
    <Posicion></Posicion>
</Informacion>
```

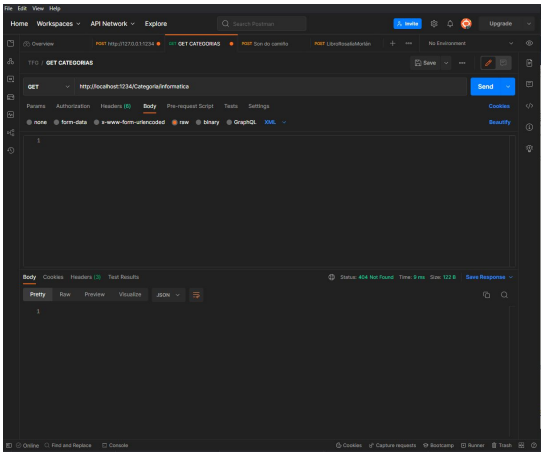


(a) Petición GET /Categoria

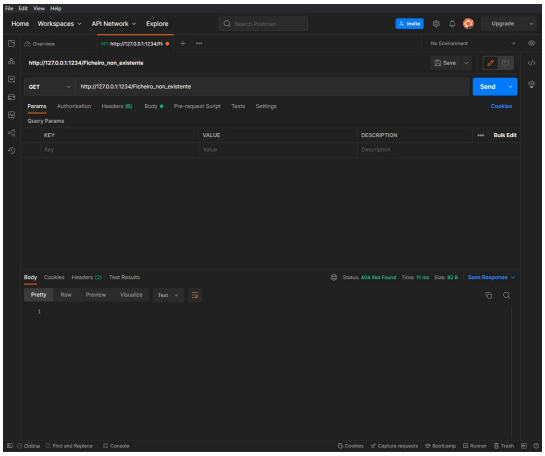


(b) Petición GET /Categoria/cultura

Figura 7.8: Peticións GET correctas



(a) Categoría ”informatica” non existe



(b) Recuperando documento inexistente

Figura 7.9: Peticións GET incorrectas

7.5 Incremento 4

Partindo do sistema obtido no incremento 2 e 3 que nos ofrece as funcionalidades do servizo, faltaranos por implementar o método de visualización dos datos. Como xa dixemos, esta parte debe adaptarse ás necesidades do cliente que vaia a usar o servizo en cada ocasión, polo que neste caso optaremos por unha versión o máis simple posible, pero intentando non perder ningunha información.

7.5.1 Seguemento e análise

Sabendo que no incremento anterior se creou un servizo capaz de enviar recursos, a implementación dun método de visualización mediante HTML, CSS e JavaScript non ten ningunha dificultade, simplemente teremos que crear unha páxina web, á cal os usuarios poderán acceder mediante o susodito servicio.

No tocante á páxina debemos analizar que características debe ter para que cumpra as necesidades do cliente. Tras facer un estudo chegouse á conclusión de que a páxina que imos deseñar e implementar debe cumprir estes obxectivos:

- Actualizarse de forma automática para mostrar as noticias máis recentes sen necesidade de ningunha acción por parte do usuario.
- Ter unha sección para cada categoría de forma separada.
- No caso da categoría ”Outras” debería especificar que categoría ten a nova, en caso de tela.

- A páxina debe autoxerarse a partir das categorías recoñecidas polo sistema. Non ten sentido que para engadir unha nova categoría haxa que refacer a páxina.

7.5.2 Deseño

Ao tratarse dun incremento fortemente ligado á visualización poderemos encontrar dous tipos de deseño: o deseño do código de JavaScript e o deseño da distribución en pantalla.

A nivel visual decidiuse que a forma máis óptima de organizar toda a información é mediante un carrusel para cada categoría que se actualizarían cada vez que acaben de mostrar as noticias; ou, o que é o mesmo, cada vez que desen unha ”volta” completa. Para intentar abarcar o máximo de información da forma máis simple posible tomáronse as seguintes decisións sobre o deseño:

- Cada carrusel crearase de forma automática usando a categoría para os identificadores dos elementos.
- Cada noticia poderá ter só unha imaxe ou vídeo, escollendo o vídeo en caso de que o JSON conteña ambos.
- Cada noticia terá o tipo de letra e cor de fondo indicado no JSON.
- As noticias da categoría ”Outras” indicarán a súa categoría cando a teñan e sexa diferente de ”outras”.
- Se o campo ”Tipo” contén a etiqueta ”Importante”, enriba da noticia aparecerá o texto ”Importante !!!” parpadeando.

Tendo en conta estas decisión podemos facer un *wireframe* como o da figura 7.10.

Figura 7.10: *Wireframe* da páxina de visualización das noticias

A nivel de código debemos decidir como funcionará o código do programa. Nun primeiro momento debemos solicitar á API cales son as categorías existentes e crear un carrusel para cada unha. Para cada carrusel débese pedir as novas da súa categoría para despois introducilas dentro e comezar co *scroll*. Unha vez o carrusel dea unha "volta" completa este baleirarase e volverá a solicitar as noticias. Ademais, para facilitar a lectura por parte dos usuarios, se se detecta o rato enriba do carrusel este bloquearase e non se moverá ata que estea fóra. Na figura 7.11 podemos ver o diagrama de secuencia dunha páxina con un único carrusel.

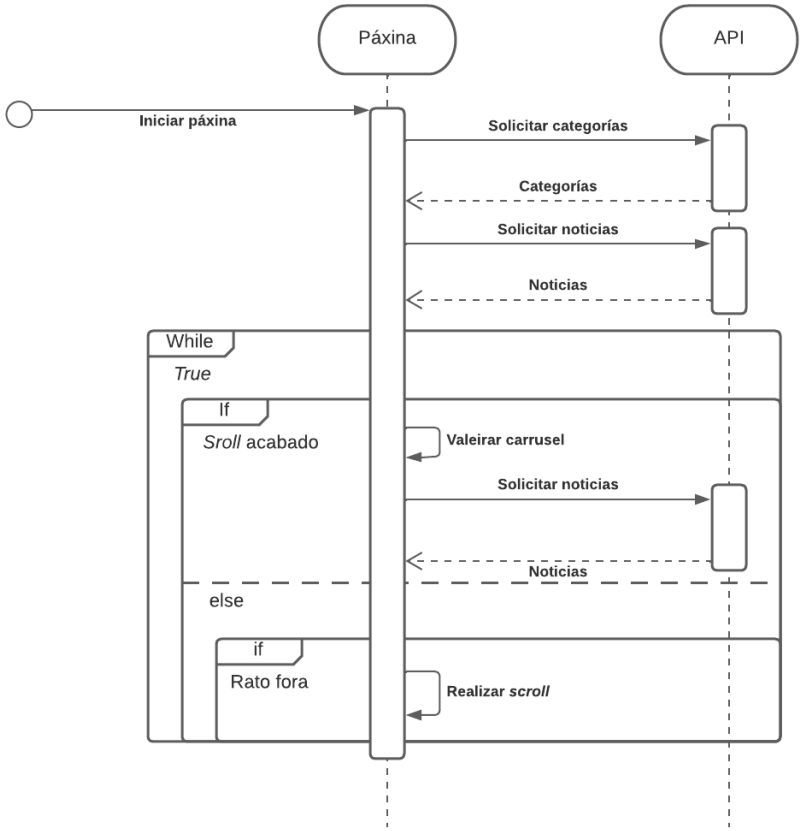


Figura 7.11: Diagrama de secuencia de unha páxina con un único carrusel

7.5.3 Desenvolvemento

Para comezar co desenvolvemento deste incremento final comezaremos creando unha páxina con un único carrusel creado a man, desta forma podemos centrarnos no bo funcionamento do carrusel, deixando de lado a formación automática da páxina tendo en conta as categorías existentes. A estrutura do carrusel será a seguinte:

```
<div class="scrollWrapper">
  <div class="scrollTitle">CATEGORIA</div>
    <div id="containerCATEGORIA" class="container">
      <div id="CATEGORIA" class="scroll">
        </div>
  </div>
</div>
```

O ”div” que ten clase ”scroll” é o corpo do carrusel, será o encargado de albergar as noticias no seu interior e ir subíndoas pouco a pouco. A clase ”container” encárgase de conter o ”scroll”, é a encargada de non permitir a visualización dos contidos cando se atopan no seu exterior.

Finalmente, como os seus nomes indican, o ”scrollTitle” é a sección que contén o título da categoría, e o ”scrollWrapper” é o encargado de unir todo (título e carrusel) nun único ”div”.

Para as noticias crearemos unha función que cree unha noticia a partir dun JSON. Esta función creará un ”div” no que irá incluíndo as diferentes partes da noticia: título, contido, imaxe ou vídeo, etc.

Unha vez temos creado o carrusel e as noticias podemos comezar coa función que lle dá movemento, pero antes crearemos unha variable chamada ”mouseOn” que conterá o carrusel sobre o que se encontra o rato actualmente. Para actualizar esta variable debemos engadir no ”scrollWrapper” os atributos ”onmouseover”, con valor ”mouseOn="CATEGORIA"”; e ”onmouseout”, con valor”mouseOn=""”. Esta variable servirá para deter o movemento do *scroll* cando o rato se atope enriba deste.

Unha vez xa temos creado este parámetro podemos comezar coa función que lle aporta o movemento ao carrusel. O funcionamento desta segue os seguintes pasos:

1. Calcúlase a altura do carrusel.
2. Colócanse as noticias na parte baixa do carrusel.
3. Se o rato non está enriba do carrusel desprázanse as noticias cara arriba.
4. Recalcúlase a altura do carrusel.
5. Compróbase se as noticias xa saíron fóra do carrusel por enriba:
    - En caso afirmativo elimínanse as noticias actuais; solicítanse as novas á API , formatéanse, insértanse e móvense á parte baixa do carrusel.
    - En caso negativo faise un *sleep* que controlará a velocidade á que baixa o *scroll*.
6. Vólvese ao paso 3.

Unha vez creado un carrusel crear varios debería ser simple; nun primeiro momento pensariamos que é suficiente con replicalos, pero atopámonos con un problema, JavaScript traballa con un único *thread*, é por iso que se decidiu usar a biblioteca ”Concurrent.Thread” [33], que nos permite simular un comportamento multifío co que si podemos replicar o código feito ata agora en múltiples carruseis.

Xuntando todo o feito ata agora e esta biblioteca, o único que nos queda por facer é unha función que reciba as categorías existentes e cree un carrusel para cada unha, cambiando aqueles sitios que poñen ”CATEGORIA” pola categoria en cuestion, e que tamén cree un fío de execución para cada un, permitindo así o *scroll* de todos os carruseis.

7.5.4 Probas

Para realizar as probas deste incremento escolleuse unha serie de noticias de diversas categorías que serían enviadas ao servizo usando Postman. Para comprobar o correcto funcionamento de todo comprobouse que todo tipo de información se mostraba correctamente, na categoría correcta e coa orde correcta. Tamén se engadiron en espazos de tempo separados para ver se se actualizaba cada vez que o scroll acababa. Os XMLs empregados son os seguintes:

```
<Informacion>
  <Fecha>25-06-2022 19:20:30</Fecha>
  <Categoria>Lexislacion</Categoria>
  <Titulo>Os mozos de 16 e 17 anos poderán conducir coches en
    2023</Titulo>
  <Texto>Os mozos de 16 e 17 anos tamén poderán conducir coches de
    forma habitual, aínda que con condicións: terán que ser
    vehículos eléctricos, cunha velocidade límite de 90 quilómetros
    por hora e un peso máximo de 400 quilogramos. Así o anunciou
    este xoves o ministro do Interior, Fernando Grande Marlaska,
    durante a presentación, xunto co director da DXT, Pere Navarro,
    da Estratexia de Seguridade Viaria 2030.</Texto>
  <Imagen>https://estaticos-cdn.prensaiberica.es/clip/
    317290bb-f626-440f-891a-cdbfed6f15e7_16-9-aspect-ratio_
    75p_1150791.jpg</Imagen>
  <Video></Video>
  <Tipo></Tipo>
  <Color>Lavender</Color>
  <Letra></Letra>
  <ColorLetra></ColorLetra>
  <Posicion>Central</Posicion>
</Informacion>
```

```
<Informacion>
  <Fecha>12-05-2022 11:08:00</Fecha>
  <Autor></Autor>
  <Categoria>Cultura</Categoria>
  <Titulo>Rosalía Morlán presenta en Santiago o libro "Camelias de
    amor"</Titulo>
  <Texto>O Auditorio Abanca de Santiago de Compostela (Rúa do
    Preguntoiro, 23) acolle este venres 6 de maio, ás 20:00 h, a
    presentación do libro ilustrado Camelias de amor, da autoría de
    Rosalía Morlán e publicado por Edicións Fervenza.

No acto, que estará aberto ao público ata completar aforo, a autora
    estará acompañada do escritor, investigador e crítico literario,
    Armando Requeixo; da concelleira de Cultura do Concello de
```

```
    Santiago de Compostela, Mercedes Rosón; do secretario xeral de
    Política Lingüística da Xunta, Valentín García; e da ilustradora
    do libro, Amalia López Brea "Luchi".

Contarase ademais co acompañamento musical de Manoele de Felisa e o
    Coro de Habaneras Rocha Forte. </Texto>
  <Imagen>https://www.noticieirogalego.com/wp-content/uploads/2022/
    05/cartel-camelias-estrada-555x391.jpg</Imagen>
  <Video></Video>
  <Tipo></Tipo>
  <Color>IndianRed</Color>
  <Letra>Lucida Handwriting</Letra>
  <ColorLetra></ColorLetra>
  <Posicion></Posicion>
</Informacion>
```

```
<Informacion>
  <Fecha>10-06-2022 12:24:38</Fecha>
  <Autor>IAGO SUAREZ</Autor>
  <Categoria>Cultura</Categoria>
  <Titulo>Derrube no palco do Son do Camiño</Titulo>
  <Texto>Ás 12.30 horas da mañá desta sexta feira, fontes do 112
    Emerxencias da Galiza confirmaban a Nós Diario o derrubamento do
    escenario principal do festival O Son do Camiño, cuxo decorrer
    está previsto para a próxima semana no Monte do Gozo en Santiago
    de Compostela.

  Varios traballadores encontrábanse no momento da caída no propio
    palco, ficando un deles atrapado na estrutura. Dous deles
    estaban en estado grave e permanecían en prognóstico reservado
    ao feche desta edición. Así o informou nun breve comunicado a
    organización do festival, no cal se indica que os servizos de
    emerxencia mobilizados evacuaron até seis persoas feridas. Os
    catro traballadores que se encontraban en estado leve foron
    dados de alta ao longo da tarde de onte.</Texto>
  <Imagen></Imagen>
  <Video>SonDoCamino.mp4</Video>
  <Tipo>Importante</Tipo>
  <Color></Color>
  <Letra></Letra>
  <ColorLetra></ColorLetra>
  <Posicion>Central</Posicion>
</Informacion>
```

```
<Informacion>
  <Fecha>11-06-2022 04:01:00</Fecha>
```

```
  <Autor>X. A. Taboada</Autor>
  <Categoria>Economia</Categoria>
  <Titulo>Goberno e Xunta abren a porta a fondos europeos á nova
    plataforma de Stellantis</Titulo>
  <Texto>Stellantis terá unha segunda oportunidade para acceder a
    fondos  europeos Next Generation cos que cofinanciar a nova
    plataforma industrial das súas instalacións en Vigo destinadas á
    fabricación de vehículos 100% eléctricos. Este proxecto quedouse
    fóra do actual Perte de automoción pola rixidez, especialmente,
    dos prazos de execución, pois as súas condicións esixen que
    todas as actuacións estivesen executadas en xuño de 2025, algo
    que resultaba imposible polo calado da reestruturación que se
    debe acometer na planta de Balaídos. Con todo, o Goberno central
    e a Xunta , conxuntamente coa multinacional, negocian que
    Stellantis poida acceder a outros fondos europeos de próximas
    convocatorias , non ao Perte actual, cos que financiar a
    remodelación das instalacións.</Texto>
  <Imagen>https://estaticos-cdn.prensaiberica.es/clip/
    b5314194-2364-4ce2-8b92-d7eb8d986917_21-9-aspect-ratio
    _75p_1151120.jpg</Imagen>
  <Video></Video>
  <Tipo></Tipo>
  <Color>SpringGreen</Color>
  <Letra></Letra>
  <ColorLetra></ColorLetra>
  <Posicion></Posicion>
</Informacion>
```

```
<Informacion>
  <Fecha>11-06-2022 10:22:35</Fecha>
  <Autor>Google</Autor>
  <Categoria></Categoria>
  <Titulo>Xá a venta o novo google chromecast</Titulo>
  <Texto></Texto>
  <Imagen></Imagen>
  <Video>http://commondatastorage.googleapis.com/gtv-videos-bucket/
    sample/ForBiggerEscapes.mp4</Video>
  <Tipo></Tipo>
  <Color>yellow</Color>
  <Letra>monospace</Letra>
  <ColorLetra></ColorLetra>
  <Posicion>Central</Posicion>
</Informacion>
```

```
<Informacion>
  <Fecha>10-06-2022 14:00:00</Fecha>
```

```
    <Autor></Autor>
    <Categoria>Deportes</Categoria>
    <Titulo>Once portos deportivos galegos reciben o distintivo de
      calidade Bandeira Azul</Titulo>
    <Texto>Once portos deportivos galegos, dez deles de titularidade
      autonómica e un estatal, veñen de ser galardoados co distintivo
      de calidade Bandeira Azul que cada ano outorga a Asociación de
      Educación Ambiental e do Consumidor (Adeac) de cara á tempada
      estival.

Nesta edición de 2021 recibiron Bandeira Azul as dársenas de
      Ribadeo, Real Club Náutico de Portosín, Real Club Náutico da
      Coruña, Sadamar, Club Náutico de Sada, Club Náutico Boiro,
      Cangas, Real Club Náutico de Sanxenxo, Nauta Sanxenxo, Monte
      Real Club de Iates de Baiona e Club Náutico de Portonovo.</Texto>
    <Imagen></Imagen>
    <Video></Video>
    <Tipo></Tipo>
    <Color></Color>
    <Letra></Letra>
    <ColorLetra>Ivory</ColorLetra>
    <Posicion></Posicion>
</Informacion>
```

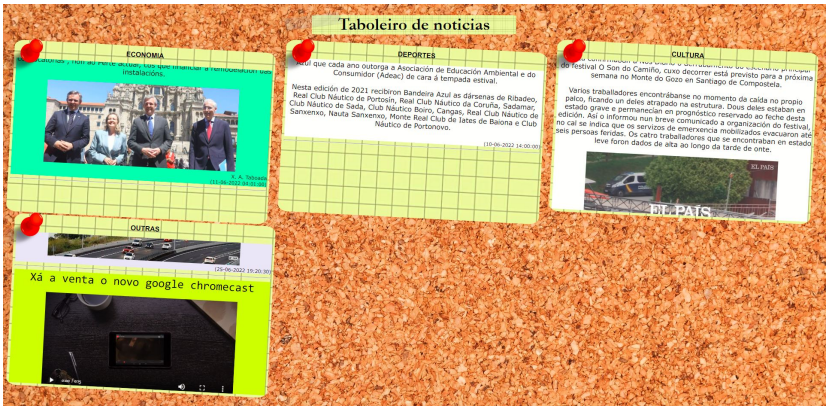


Figura 7.12: Páxina final en funcionamento

7.6 Incremento 5

Finalmente, baseándonos na páxina creada no incremento anterior imos deseñar dúas páxinas de maior complexidade que satisfaga as necesidades do cliente.

7.6.1 Seguemento e análise

Unha vez finalizado a páxina do incremento anterior xa se ten un coñecemento de como tratar coa información proporcionada pola API, polo que se decidiron crear dúas páxina de maior complexidade, unha que separe as noticias de unha que imite un periódico no que o usuario vai pasando as páxinas e outra que imite a composición creada para un telexornal . Para que cumpra cos requisitos do cliente debe cumprir o seguinte:

- Actualizarse de forma automática para mostrar as noticias máis recentes sen necesidade de ningunha acción por parte do usuario cada vez que se acaben as noticias que se teñen en memoria.
- Todas as noticias deben ir acompañadas da súa categoría.
- No caso da páxina que funciona como un periódico poderase escoller entre as diversas categorías para ver as súas noticias.
- No caso da páxina que imita un telexornal contará cunha zona central para as noticias importantes e unha zona para o resto de noticias.

7.6.2 Deseño

Dentro do deseño desta páxina, ao igual que no incremento anterior, poderemos atopar dúas partes dentro do deseño: o deseño do código de JavaScript e o deseño da distribución en pantalla.

Periódico

No tocante a visualización existirán dúas partes: a zona de categorías, onde o usuario pode escoller a categoría; e a zona coa noticia. Tendo en conta este formato podemos estipular o seguinte:

- A zona para escoller categorías será un menú na parte alta da pantalla.
- A cor indicada no JSON usarase para darlle fondo ao título.
- As imaxes aparecerán ao comezo da noticia a excepción de se teñen vídeo, nese caso incluiranse ao final.
- A páxina só pode avanzar, asegurándose así que o usuario acabe as noticias e volva a solicitar as novas noticias.

Tendo en conta esta características podemos realizar o *wireframe* da figura 7.13.

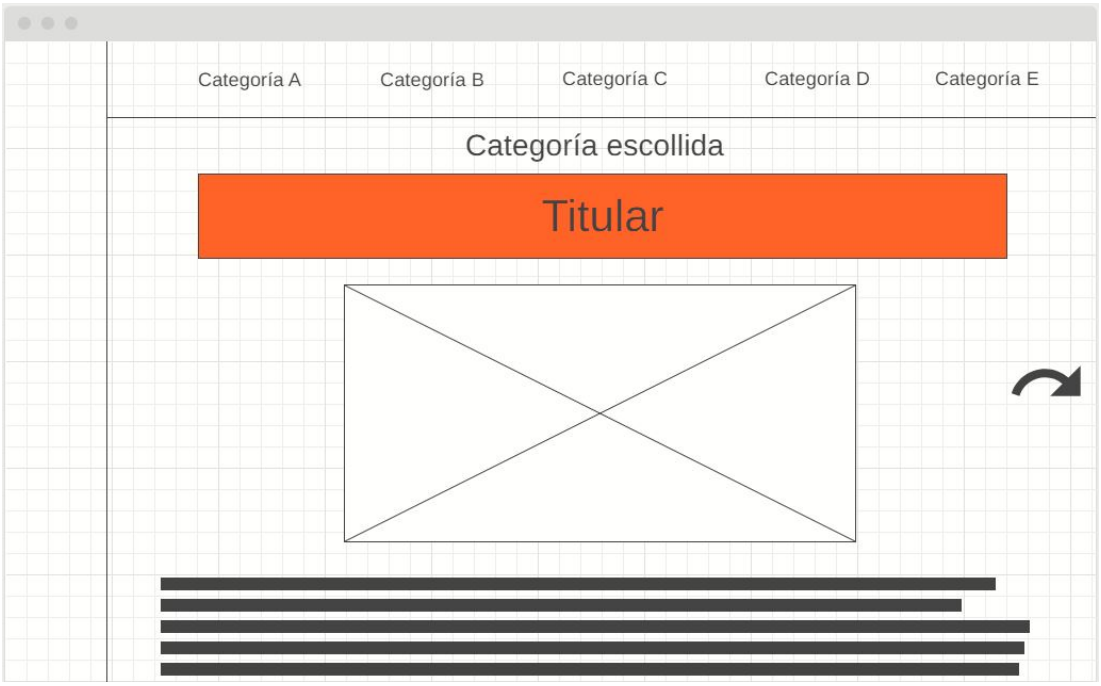


Figura 7.13: *Wireframe* da páxina que imita un periódico

A nivel de código o funcionamento pódese reducir ás mesmas dúas zoas, dando lugar a dous comportamento diferentes.

A zona das noticias constará de unha lista de noticias da categoría seleccionada que mostrará de forma ordenada a medida que o usuario avance. Cada vez que a lista se acabe solicitará ao servizo unha nova lista de noticias.

No caso das categorías será unha sección que se cree de forma automática ao iniciar a páxina e que contén o nome de todas as categorías. Cando o usuario fai clic nunha destas categorías a lista de noticias substitúese por unha lista da categoría seleccionada e forzase á páxina a avanzar.

Telexornal

Para a visualización xa se especificou que hai dúas zonas principais da pantalla, unha zona para as novas importantes e outra para o resto. Partindo desta base decidíronse os seguintes puntos:

- A zona de noticias non importantes será unha barra de información na zona baixa da pantalla, mentres que as importantes ocuparan todo o resto.

- No caso das noticias importantes que conteñan un vídeo no JSON simplemente se escribirá a categoría e o título mentres se reproduce o vídeo, e en caso contrario escribirase o corpo da noticia durante un tempo predeterminado.
- Para as noticias non importantes so se mostrará categoría e título.
- Cada noticia non importante terá o cor de fondo indicado no JSON como o cor de fondo da categoría.
- Considéranse noticias importantes aquelas cuxo campo ”Posicion” contén a etiqueta ”Central”.

Tendo en conta estas decisión podemos facer un *wireframe* como o da figura 7.14.

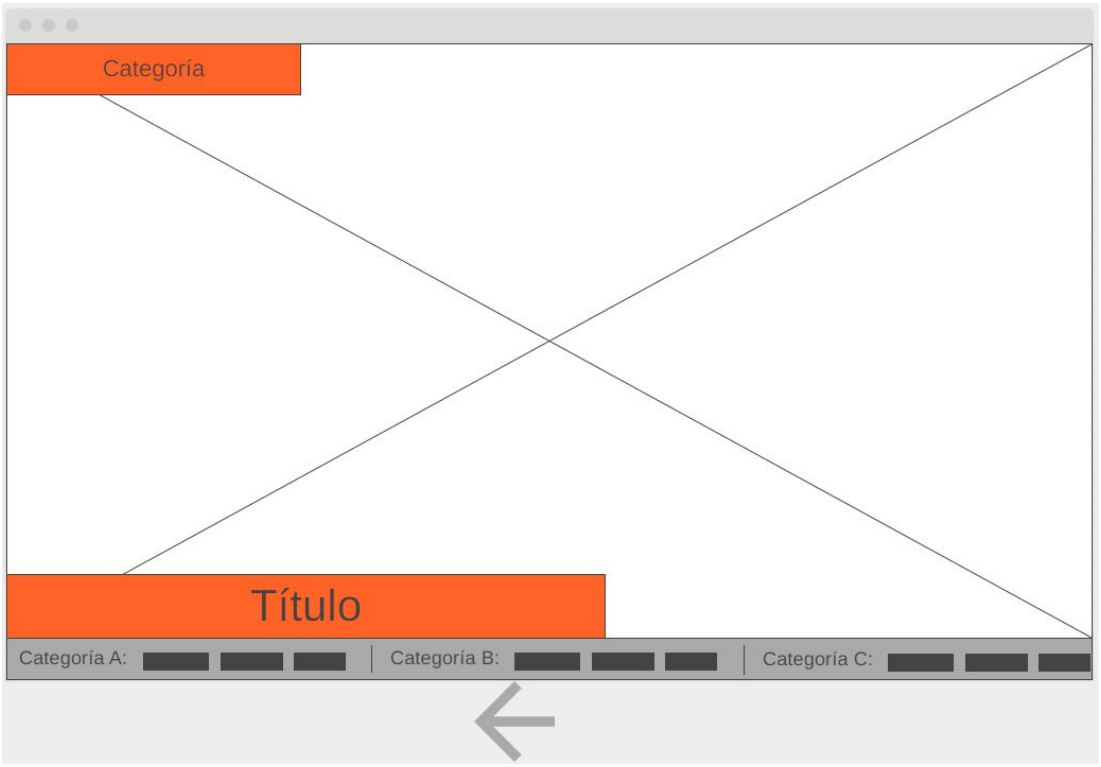


Figura 7.14: *Wireframe* da páxina que imita un telexornal

A nivel de código debemos decidir como funcionará o código do programa, tendo en conta que as noticias importantes e non importantes terán comportamentos diferentes.

No caso das non importantes debemos solicitar á API cales son as categorías existentes no servizo, para de seguido engadir todas as novas de cada categoría que non sexan importantes e comezar o *scroll*. Unha vez o carrusel dea unha ”volta” completa este baleirarase e volverá a solicitar as noticias.

Para as importantes tamén comezará solicitando as categorías. Unha vez ten a lista comezará solicitando de forma ordenada as novas de cada unha das categorías para atopar aquelas que son importantes. Se atopa unha noticia importante con vídeo reprodúceo, senón pon o corpo da noticia por pantalla durante un tempo determinado. Finalmente, tras revisar todas as categorías volve a comezar seguindo a mesma orde.

7.6.3 Desenvolvemento

Periódico

A primeira parte a abordar é a creación do menú de selección de categoría da zona superior. Para crear esta parte a páxina solicita á API a lista de categorías. Unha vez temos esta lista creamos a sección, facendo que cada categoría funcione como un botón que ao pulsalo solicite as noticias desa categoría e mostre a primeira.

Para a zona na que se mostran as noticias terá a seguinte funcionalidade:

1. Ao inicio da execución terá unha pantalla principal e o botón de pasar páxina estará oculto, pois non hai ningunha categoría seleccionada.
2. Esperase ata que usuario seleccione a categoría e se consiga unha lista de noticias.
3. Mostrase a primeira noticia non vista da lista.
4. Esperase a que o usuario toque algún botón, podendo darse dous casos:
    - Se o usuario pulsa unha categoría a páxina solicita unha lista nova desa categoría e volve ao paso 3.
    - Se o usuario pasa á seguinte noticia sigue coa execución.
5. Se a lista está baleira solicitase unha nova lista de noticias.
6. volvese ao paso 3.

Cabe destacar que se unha noticia xa lida volve a vir nunha lista diferente considerase non lida.

Telexornal

Para comezar co desenvolvemento desta páxina comezaremos separando o espazo da páxina en dúas zonas: a inferior ,onde aparecerán as noticias non importantes, e a central, onde aparecerán as noticias importantes

As noticias non importantes teñen unha función que cree un ”div” a partir dun JSON. Este ”div” conterá a noticia que ten que aparecer na zona baixa, coa súa categoría e título.

Para as noticias importantes teremos unha función encargada de, por un lado, editar o titular e categoría que aparecen na zona central, e por outro lado, mostrar a noticia da forma adecuada segundo teña vídeo ou non.

Unha vez temos creado o carrusel e as noticias non importantes podemos comezar coa función que lle dá movemento:

1. Calcúlase a lonxitude do carrusel.
2. Colócanse as noticias na parte dereita do carrusel.
3. Recalcúlase a altura do carrusel.
4. Compróbase se as noticias xa saíron fóra do carrusel pola esquerda:
    - En caso afirmativo elimínanse as noticias actuais; solicítanse as novas á API , formatéanse, insértanse e móvense á parte dereita do carrusel.
    - En caso negativo realízase un *sleep* que controlará a velocidade á que se despraza o *scroll*.
5. Vólvese ao paso 3.

Para o funcionamento da zona central os pasos son os seguintes:

1. Escóllese unha categoría que aínda non se comprobase. En caso de non habela márcanse todas as categorías como non comprobadas.
2. Ecóllese unha noticia importante aínda non escollida da categoría seleccionada no paso 1. En caso de non habela volvese ao paso 1.
3. Escríbense o título e categoría da noticia.
4. Dependendo dos contidos da noticia poden pasar dúas cousas:
    - Se a noticia ten vídeo, este reprodúcese de forma automática ata que este finalice.
    - Se non ten vídeo, mostrase por pantalla o corpo da noticia durante un tempo predeterminado.
5. Vólvese ao paso 2.

Xuntando estas dúas funcións conseguimos que a nosa páxina mostre toda a información separando aquelas novas que son relevantes de aquelas que non o son, dando lugar a unha presentación dinámica e agradábel ao usuario.

7.6.4 Probas

No tocante ás probas usáronse os mesmos XML que no incremento anterior.

Para a páxina que imita un periódico comprobaremos como ao cambiar de categoría a páxina recupera un novo grupo de novas, e cada vez que se ven todas as noticias dunha categoría volvense a solicitar.

No caso da que semella un telexornal, con estas novas comprobamos como aquelas noticias marcadas como ”Central” no campo ”Posicion” se mostran na parte de maior tamaño da páxina, mostrando o vídeo ou o corpo da nova segundo conveña. Tamén se comproba que as novas non importantes se vexan na zona baixa da páxina, mostrando a categoría e o título no carrusel.

Podemos ver unha mostra de como quedarían estan páxinan nas figuras 7.15 e 7.16.

Figura 7.15: Páxina final do incremento 5 en funcionamento que imita un periódico

Figura 7.16: Páxina final do incremento 5 en funcionamento que imita un telexornal

Capítulo 8

Conclusións e liñas futuras

UNHA vez finalizado esta proxecto, conseguimos un sistema capaz de funcionar como unha API e un servidor de recursos ao mesmo tempo, permitíndonos xerar un panel informativo en tempo real. A continuación analizaremos algúns aspectos do proxecto e o cumprimento dos obxectivos propostos inicialmente.

Dentro das características principais atopábase acadar a autoorganización das noticias. Neste aspecto o sistema logra este obxectivo mediante o ”manager”, que as clasifica por categoría e ordena por prioridade. A única situación irresoluble sería a recepción de datos erróneos.

Outro punto crítico era a adaptabilidade. Como xa se dixo ao longo do proxecto, existe un servizo central que busca ter todas as súas funcionalidades independentes da entrada e saída que usa o usuario. Isto foi posible grazas ao uso de interfaces no caso de ”Informacion” e do *parser*, de tal xeito que calquera conxunto de clases que implementen esas dúas interfaces poderán ser usadas para analizar a entrada. Tamén cabe destacar que a nivel visualización a estrutura de ficheiro HTML que se conecta coa API usando JSONs permite unha independencia completa entre vista e servizo.

Tamén é relevante que a función de servidor que desempeña o servizo permita gardar os recursos que se precisen en todo momento, non sendo necesario ningún outro sistema adicional.

Finalmente, un dos obxectivos máis importantes era a visualización en tempo real. Cada vez que o servizo central recibe un novo elemento de información, este engádese á categoría que corresponde podendo ser solicitado de xeito inmediato á súa dispoñibilidade.

En conclusión, neste proxecto cumpríronse os obxectivos de autoorganización, adaptabilidade e autosuficiencia. No tocante á resposta en tempo real temos un pequeno conflito coa adaptabilidade; ao permitir crear novos métodos de visualización personalizados que se adapten ao gusto do usuario, non temos xeito de asegurarnos de que estas novas páxinas creadas polo cliente soliciten información máis recente de forma periódica.

8.1 Conclusións da aprendizaxe persoal

Trátase dun proxecto que a primeira vista parece sinxelo, pero realmente é un sistema que mestura múltiples tecnoloxías que abarcan diferentes campos, dando lugar a un traballo que require empregar coñecementos de diversas materias estudadas ao longo do grao.

No analizador léxico-sintáctico foi necesario o coñecemento que se tiña de deseño e creación de gramáticas e linguaxes, usando ferramentas aprendidas en materias como ”Procesamento de Linguaxes”,”Teoría da Computación” e ”Matemática Discreta”; onde se aprendeu sobre as gramáticas e o seus analizadores, autómatas necesarios para a creación de *parsers* ou o uso de Flex e Bisson.

A API requeriu coñecementos sobre os protocolos HTTP e a creación de servizos; coñecementos que podemos atopar en materias como ”Redes”, ”Internet e Sistemas Distribuídos” ou ”Marcos de Desenvolvemento”; onde aprendemos a facer servizos de acceso a recursos, de uso para páxinas web ou de acceso a bases de datos.

Para a parte visual precisáronse coñecementos de deseño web, como os aprendidos en ”Interfaces Persoa Máquina” ou ”Marcos de Desenvolvemento”; onde se aprendeu o uso de HTML, CSS, JavaScript e JSON, así como o proceso de deseño de páxinas web multidispositivo con uso de código.

Finalmente, cabe destacar que para a organización do proxecto e a redacción desta memoria usáronse coñecementos de materias como ”Deseño Software”, ”Proceso Software”, Xestión de Proxectos” ou ”Deseño de Sistemas Intelixentes”, onde aprendemos a realizar a análise de requisitos, representar multitude de diagramas (Gantt, clases, secuencia, etc), usar metodoloxías e todo o que sabemos relacionado con xestión de proxectos.

8.2 Liñas futuras

Está claro que este proxecto propón unha base estable para a creación dun sistema de visualización de información o máis universal posible, pero hai certos aspectos que poderian ser contrastados con outras implementacións para comprobar cales son máis beneficiosas.

- Pode ser de gran interés estudar a posibilidade de crear un *parser* de XML universal que gardase todos os campos. Esta podería ser unha boa idea, pois implicaría unha carga de traballo moito menor á hora de adaptar o sistema a novas contornas, pero tamén sería preciso estudar como afectaría ao funcionamento do servizo; pois ao permitir calquera campo necesitaríamos mapear o nome de cada un co seu valor, así como levar rexistro de todos os campos lidos para cada noticia, o que podería supoñer un aumento nas necesidades de recursos para almacenamento e computación.

- Para o problema comentado na conclusión de que non se pode controlar se a páxina final ten tempo real ou non, hai unha solución, que a API devolva fragmentos de JavaScript e HTML no lugar de JSONs, de tal forma que forcemos á paxina a funcionar en tempo real. De todas as maneiras, esta solución non melloraría o noso sistema, simplemente o faría diferente, pois a cambio de asegurarnos esa resposta en tempo real estamos sacrificando parte da adaptabilidade que ten o proxecto.
- Por suposto, tal e como se dixo ao comezo desta sección, este proxecto é unha base, polo que se pode partir del para crear sistemas máis complexos con ”managers” con máis funcionalidades, un mellor sistema de seguridade, que conteña varias APIs a un mesmo tempo,etc.

Apéndices

Apéndice A

Material adicional

A.1 Códigos do protocolo HTTP

Codigo	Nome	Significado
200	*OK*	Resposta estándar para peticións correctas
201	*Created*	Petición completada correctamente resultando na creación dun novo recurso.
400	*Bad Request*	O servidor non procesará a petición por un erro do cliente. No caso deste proxecto un erro de sintaxe no XML.
403	*Forbidden*	Solicitude rexeitada por ser de un cliente sen permisos.
404	*Not Found*	O recurso non foi atopado ou non existe.

Táboa A.1: Códigos do protocolo HTTP utilizados ao longo deste proxecto

A.2 Uso do servizo

Para o uso deste servizo non se require instalación do programa, simplemente necesitaremos un equipo que teña unha versión de Java compatible.

Unha vez o usuario teña o sistema descargado no seu equipo o primeiro que debe facer é configurar o arquivo ”properties.txt” cos parámetros adecuados. Un exemplo de arquivo correcto é o que podemos ver no *listing* A.1. Para usar as páxinas feitas neste traballo o parámetro ”DIR” debe coincidir coa carpeta ”/Code/HttpDocs” que existe dentro do proxecto, pero non é obrigatorio.

```
1 PORT=1234
2 DIR=D:/HttpDocs
3 CATEGORIAS=Economia;Deportes;Cultura
4 CATEGORIASIZE=5
5 IPPOST=0:0:0:0:0:0:0:0-0:0:0:0:0:0:0:f;122.0.0.2-133.0.0.0
```

Listing A.1: Ficheiro ”Properties.txt” de exemplo.

Co arquivo xa configurado so temos que executar o sistema e este xa estará listo para utilizar. Para executar o sistema simplemente teremos que compilar a clase principal ”ServiceMain.java” que se atopa en ”/src/es/udc/InMa/service” dende a carpeta ”src” usando ”javac”, o compilador de Java. Finalmente executaremos o arquivo resultante da compilación mediante o comando ”java”.

Para introducir información podemos usar calquera sistema ou aplicación que permita realizar peticións POST, como por exemplo Postman.

No caso de querer usar as páxinas de exemplo que se crearon neste TFG as súas rutas son:

- ”/index.html” para o taboleiro de chinchetas.
- ”/index2.html” para a páxina que semella un telexornal.
- ”/index3.html” para a páxina que semella un periódico.

Relación de Acrónimos

API Application Programming Interface. 5, 25, 26, 40, 49, 51, 56, 58–60, 63–65

CSS Cascading Style Sheets. 14, 26, 27, 40, 47, 64

CUP Construction of Useful Parsers. 12, 13, 27, 28

HTML HyperText Markup Language. 13–15, 26, 27, 32, 35, 40, 47, 63–65

HTTP Protocolo de transferencia de hipertexto. iii, vi, 12, 16, 27, 28, 38, 39, 42, 64, 67

IDE Integrated Development Environment. 16, 27

JSON JavaScript Object Notation. 5, 13, 15, 26, 27, 29, 32, 35, 38, 39, 44, 48, 51, 56, 58, 59, 63–65

PDF Portable Document Format. 17

PL Procesamento de Linguaxes. 27

RSS Really Simple Syndication. iv, 4, 9, 10

W3C World Wide Web Consortium. 14, 15

XML eXtensible Markup Language. 2, 5, 15, 26, 27, 30, 31, 35, 38, 45, 52, 61, 64, 67

YACC Yet Another Compiler-Compiler. 13

Bibliografía

[1] U. D. C. Facultade de Ciencias da Comunicación, "Grao en comunicación audiovisual," consultado o 2022-06-28. [En liña]. Dispoñible en: https://comunicacion.udc.es/gl/comunicacion-audiovisual

[2] U. D. Coruña, "Detalle do grao en ciencia e enxeñaría de datos," consultado o 2022-06-28. [En liña]. Dispoñible en: https://estudos.udc.es/gl/study/detail/614g02v01

[3] S. R. Pérez, "Guía de uso de la herramienta de diseño gráfico canva," consultado o 2022-06-28. [En liña]. Dispoñible en: https://support.google.com/blogger/?p=help_home&hl=gl&authuser=0#topic=3339243

[4] Google, "Centro de axuda de blogger," 2020, consultado o 2022-06-28. [En liña]. Dispoñible en: https://bibliosaude.sergas.gal/DXerais/864/GUIA_CANVA.pdf

[5] A. E. B. O. del Estado, "Ley 21/2014, de 4 de noviembre, por la que se modifica el texto refundido de la ley de propiedad intelectual, aprobado por real decreto legislativo 1/1996, de 12 de abril, y la ley 1/2000, de 7 de enero, de enjuiciamiento civil." in *«BOE» núm. 268, de 5 de noviembre*, 2014, consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2014-11404

[6] ——, "Real decreto-ley 24/2021, de 2 de noviembre, de transposición de directivas de la unión europea en las materias de bonos garantizados, distribución transfronteriza de organismos de inversión colectiva, datos abiertos y reutilización de la información del sector público, ejercicio de derechos de autor y derechos afines aplicables a determinadas transmisiones en línea y a las retransmisiones de programas de radio y televisión, exenciones temporales a determinadas importaciones y suministros, de personas consumidoras y para la promoción de vehículos de transporte por carretera limpios y energéticamente eficientes." in *«BOE» núm. 263, de 3 de noviembre*, 2021, consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.boe.es/buscar/act.php?id=BOE-A-2021-17910

[7] R. M.-B. Asensio, "¿qué es rss?" 2018, consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.um.es/docencia/barzana/PRACTICAS/RSS-Google-Reader.html

[8] Liferea, "Liferea documentation," consultado o 2022-06-28. [En liña]. Dispoñible en: https://lzone.de/liferea/docs.htm

[9] Feedly, "How to use feedly," consultado o 2022-06-28. [En liña]. Dispoñible en: https://blog.feedly.com/get-the-right-content-on-your-feedly/

[10] Adobe, "Adobe premiere pro," consultado o 2022-06-28. [En liña]. Dispoñible en: https://helpx.adobe.com/es/pdf/premiere_pro_reference.pdf

[11] A. T. S.L., "Información sobre showscreen," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.showscreen.es/showscreen/uploads/sites/69/2019/06/Informacion_Showscreen.pdf

[12] Oracle, "Documentación java ee," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.oracle.com/es/java/technologies/java-ee-glance.html

[13] ——, "Httpserver manual," consultado o 2022-06-28. [En liña]. Dispoñible en: https://docs.oracle.com/javase/8/docs/jre/api/net/httpserver/spec/com/sun/net/httpserver/HttpServer.html

[14] A. Commons, "Commons-text manual," consultado o 2022-06-28. [En liña]. Dispoñible en: https://commons.apache.org/proper/commons-text/

[15] ——, "Commons-lang manual," consultado o 2022-06-28. [En liña]. Dispoñible en: https://commons.apache.org/proper/commons-lang/

[16] G. Klein, S. Rowe, and R. Décamps, "Jflex user's manual," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.jflex.de/manual.pdf

[17] S. Mittal, "Flex (fast lexical analyzer generator)," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.geeksforgeeks.org/flex-fast-lexical-analyzer-generator/

[18] T. Hurka, "Byacc/j home," consultado o 2022-06-28. [En liña]. Dispoñible en: http://byaccj.sourceforge.net/

[19] M. Petter, S. E. Hudson, G. Visualization, and G. I. o. T. Usability Center, "Cup user's manual," consultado o 2022-06-28. [En liña]. Dispoñible en: http://www2.cs.tum.edu/projects/cup/docs.php

[20] WHATWG, "Html living standard," consultado o 2022-06-28. [En liña]. Dispoñible en: https://html.spec.whatwg.org/multipage/

[21] W. W. W. Consortium, "Html & css," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.w3.org/standards/webdesign/htmlcss

[22] ——, "Javascript web apis," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.w3.org/standards/webdesign/script

[23] ——, "Extensible markup language (xml) 1.1 (second edition)," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.w3.org/TR/xml11/

[24] IBM, "Formato json (javascript object notation)," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format

[25] E. Foundation, "Eclipse documentation," consultado o 2022-06-28. [En liña]. Dispoñible en: https://help.eclipse.org/latest/index.jsp

[26] I. Postman, "Postman learning center," consultado o 2022-06-28. [En liña]. Dispoñible en: https://learning.postman.com/docs/getting-started/introduction/

[27] I. GitHub, "Github docs," consultado o 2022-06-28. [En liña]. Dispoñible en: https://docs.github.com/es

[28] L. P. Team, "The latex project, core documentation," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.latex-project.org/help/documentation/

[29] Overleaf, "Overleaf documentation," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.overleaf.com/learn

[30] K. Schwaber and J. Sutherland, "The defnitive guide to scrum: The rules of the game," 2017, consultado o 2022-06-28. [En liña]. Dispoñible en: https://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[31] A. E. B. O. del Estado, "Resolución de 7 de octubre de 2019, de la dirección general de trabajo, por la que se registra y publica el xix convenio colectivo del sector de empresas de ingeniería y oficinas de estudios técnicos." in *«BOE» núm. 251, de 18 de octubre*, 2019, consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.boe.es/diario_boe/txt.php?id=BOE-A-2019-14977

[32] I. Free Software Foundation, "Gnu bison," consultado o 2022-06-28. [En liña]. Dispoñible en: https://www.gnu.org/software/bison/

[33] L. D. Seta, "Programación multihilos con javascript)," consultado o 2022-06-28. [En liña]. Dispoñible en: https://dosideas.com/noticias/java/136-programacion-multihilos-con-javascript