



UNIVERSIDADE
DE VIGO

ESCOLA SUPERIOR DE ENXEÑARÍA INFORMÁTICA

Memoria do Traballo de Fin de Grao que presenta

D. Adolfo Álvarez López

para a obtención do Título de Graduado en Enxeñaría Informática

Deseñador Gráfico de interfaces de usuario para Mono/Winforms



Xullo, 2014

Traballo de Fin de Grao Nº: EI 13/14 - 039

Titor/a: Dr. Baltasar García Pérez Schofield

Área de coñecemento: Linguaxes e Sistemas Informáticos

Departamento: Informática

Agradecementos

O meu titor de proxecto Baltasar García Pérez-Schofield pola súa cercanía e a súa dedicación e tempo invertido.

A miña familia e amigos pola seu gran apoio e axuda prestadas, en especial a Nancy, Andrés, Abel, Rober, Mauro, Marcos e Alberto que en todo momento estiveron dispostos axudarme en todo o que lles fora posible.

Moitas grazas.

Memoria

1. Introducción	3
1.1. Identificación do proxecto	3
1.2. Organización da documentación	3
1.3. Marco da aplicación	4
2. Obxetivos da aplicación	5
3. Descrición técnica da aplicación	6
3.1. Winforms vs Windows Presentation Foundation vs GTK#	8
3.2. XML vs XAML	9
3.3. NUnit	10
3.4. Git	10
4. Metodoloxías utilizadas	10
5. Planificación e presuposto	13
5.1. Planificación	13
5.2. Presuposto	19
5.2.1 Custo do hardware	19
5.2.2. Custo do software	20
5.2.3 Custo do persoal do proxecto	21
5.2.4 Custo total do proxecto	21
6. Problemas atopados e solucións aportadas	21
7. Futuras ampliacións	22
8. Bibliografía	23

Manual Técnico

1. Introducción	27
2. Investigación preliminar	28
3. Primer prototipo	34
3.1. Diseño rápido.....	34
3.2. Construcción do prototipo.....	45
3.3. Evaluación e retroalimentación do prototipo.....	46
4. Segundo prototipo	46
4.1. Diseño rápido.....	46
4.2. Construcción do prototipo.....	52
4.3. Evailuación e retroalimentación do prototipo.....	53
5. Tercer prototipo	54
5.1. Diseño rápido.....	54
5.2. Construcción do prototipo.....	57
5.3. Evaluación e retroalimentación do prototipo.....	58
6. Desenvolvimento do producto.....	58

Manual de Usuario

1. Requerimentos Hardware e Software.....	63
2. Execución	6464
3. Funcionamento	72
3.1. A ferramenta de deseño.	744
3.1.1. A interfaz	744
3.1.2. O formulario.....	755
3.1.3. Os elementos	76
3.2. Exportar e importar interfaces.....	988
3.2.1. Exportar.....	99
3.2.2 Importar.....	99
4. Posibles erros.....	100
4.1. Non ter instalado o Software preciso	100
4.2. Erro ó abrir ou importar unha interfaz	101

MEMORIA

1. Introducción

1.1. Identificación do proxecto

Título: Deseñador gráfico de interfaces de usuario, Winforms.

Código: EI 13/14 - 039

Autor: Adolfo Álvarez López

DNI: 53199109V

Curso: 2013-2014

Titor do TFG: Baltasar García Pérez Schofield

Cotitora: Lourdes Borrajo Diz

Área de Linguaxes e Sistemas Informáticos

Departamento de Informática

Universidade de Vigo

1.2. Organización da documentación

Memoria: Documento formal no que se detallan todas as etapas seguidas no proceso de desenvolvemento do sistema incluíndo os contratempos sufridos e outras dificultades, as cales quedarán reflexadas na comparativa entre representación temporal do traballo e a planificación inicial.

Manual técnico: Documento formal no cal se incluírá toda a información técnica do proxecto para facilitar as futuras labores de mantemento do sistema

Manual de usuario: Documento informal no cal se fará unha explicación ó usuario de cómo instalar e utilizar a ferramenta entregada.

1.3. Marco da aplicación

Na actualidade o uso dos sistemas operativos de microsoft (desde windows XP ata Windows 8) e superior o 80% polo que non é de estrañar que se pretenda desenvolver software para esta plataforma.

O desenvolvemento de aplicacións para este sistema operativo baséase no Framework .NET, este inclúe as linguaxes de programación c# y VisualBasic, el Common Language Runtime e unha gran biblioteca de clases. O equivalente a .NET de código aberto é Mono, o cal permite desenvolver aplicacións para .NET noutras plataformas.

Nestes frameworks existen diversas APIS para a creación de interfaces gráficas, pero teñen en común a API Winforms. Esta foi creada para .NET pero co paso dos anos Mono foi capaz de emulala completamente. Winforms permite crear interfaces mediante instrucións de código no interior do código dun proxecto. Pero esta API ten dous problemas, o primeiro é a dificultade que existe para separar a parte de lóxica dunha aplicación da parte da vista que correspondería ó uso dos Winforms, e o segundo é que ó difícil que pode ser nalgúns casos obter a interface tal é como a deseñamos. En interfaces un pouco complexas que se compoñen de diversos elementos diferentes, posicionar e dimensionalos da maneira que pensáramos pode ser difícil mediante as instrucións proporcionadas pola API, baseadas todas en valores numéricos e coordenadas, o que fai que se teña que probar varias veces o código correspondente á interface para asegurarse de que se mostra tal e como se planea.

Debido a estas dificultades creemos que se vota de menos una ferramenta visual que permita deseñar, crear, posicionar e dimensionar os elementos que compoñen a nosa interface sen necesidade de ter que xogar con diferentes combinacións de instrucións, valores e coordenadas directamente sobre o código, se non que te permita ir vendo como a constrúes pouco a pouco hasta obter o que realmente queres sen precisar de moito tempo.

Visual Studio, un IDE desenvolto por Microsoft, para programar nos linguaxes soportados por .NET, conta con unha ferramenta que cubre esta necesidade. Pero presenta un problema importante, tan só está dispoñible para Windows, non é libre e precisa de licencia. En Mono tamén se propón o uso dunha ferramenta que si que é libre e esta dispoñible para linux pero

tras probala nos damos de conta que o engadir poucos elementos ou en certos ordenes a aplicación deixa de funcionar, explicación para isto podería ser que non se actualiza dende xa fai tres anos.

Neste proxecto propónse a creación dunha aplicación multiplataforma (Windows e GNU/Linux), visual e de uso sinxelo que permita deseñar e construír interfaces gráficas da API Winforms e exportalas para logo podelas usar noutro proxecto de desenvolvemento.

2. Obxectivos da aplicación

A finalidade principal deste proxecto é a creación dunha aplicación libre para o deseño de interfaces de usuario para proxectos baseados en .NET ou Mono.

A aplicación está orientada para ser utilizada por desenvolvedores e que lles permita, dunha forma rápida e sinxela xerar interfaces gráficas para podelas usar nos seus proxectos. Por isto buscarase que a aplicación teña unha interface sinxela e directa para que o seu uso sexa o mais natural posible.

A interface comporase de tres áreas:

- Un treeview lateral no que se poderá observar de forma esquemática o conxunto de elementos que compoñen a interface que se está a desenvolver.
- Un datagridview no outro lateral, onde estará o listado dos atributos dos elementos engadidos, que se poderán modificar para darlle forma os compoñentes da interface.
- Unha zona de traballo central onde poderase observar como se vai construíndo a interface en desenvolvemento.

Esta ferramenta permitirá crear interfaces gráficas sinxelas cós elementos mais utilizados da API Winforms. Entre eles destacamos os seguintes:

- Paneles contenedores:
 - Border: Panel básico da API Winforms
 - Grid: TableLayoutPanel da API Winforms

- HBox e VBox: Casos especiais de TableLayoutPanel. No primeiro os elementos que se lle engadan o farán horizontalmente, mentres que no segundo caso será verticalmente.
- TabControl, GroupBox.
- Menús:
 - MenuStrip
 - ToolBar
- Controles:
 - Button, Label, TextBox, PictureBox, DateTimePicker, MonthCalendar, RadioButton, CheckBox, ProgressBar, Splitter
 - Combobox, TreeView, StatusBar
 - DataGridView, ListView

Aparte de poder deseñar os controles, paneles e menús soportados, tamén se permitirá crear ós eventos mais importantes para estes elementos. Estes serían:

- Click, DoubleClick
- Enter, Leave, GotFocus, LostFocus
- KeyDown, KeyPress, KeyUp
- MouseClick, MouseDoubleClick, MouseWheel, MouseDown, MouseUp, MouseEnter, MouseLeave, MouseHover
- Resize

Ademais, aparte da ferramenta que permitirá o deseño e exportación das interfaces, preténdese crear unha biblioteca que permita importar estas interfaces no proxecto no que se precisen utilizar, e así mediante unha simple función, poder utilizar a interface aforrando todo o código correspondente á vista da aplicación.

3. Descripción técnica da aplicación

O sistema desenvolvido deberá permitir a creación de interfaces gráficas da a API Winforms

de .Net e Mono, dando a posibilidade de exportalas e importalas noutros proxectos en desenvolvemento. A API Winforms úsase en Visual Basic e C# e será neste último no que se desenvolverá o noso sistema.

O sistema comporase de dúas partes:

- **Ferramenta de deseño e creación de interfaces:** Codificada en C# e con ela poderase crear graficamente a interface que se desexe.
- **Biblioteca .dll:** Con esta biblioteca engadida como referencia nun proxecto C# poderase importar e utilizar a interface deseñada coa ferramenta de deseño.

Na figura 1 móstrase a arquitectura do sistema proposto. A ferramenta xeradora de Winforms terá os propios Winforms formando a capa de presentación da aplicación. Como xa se mencionou a lóxica completa será creada en C# e por último a persistencia da aplicación será en ficheiros Xml. Estes últimos ficheiros son os que se poderán importar nunha aplicación en construción mediante o uso da biblioteca que tamén se aportará, e que creará a interface en Winforms constituíndo a capa de presentación da nova aplicación, tal e como se indica no bloque da dereita da Figura 1.

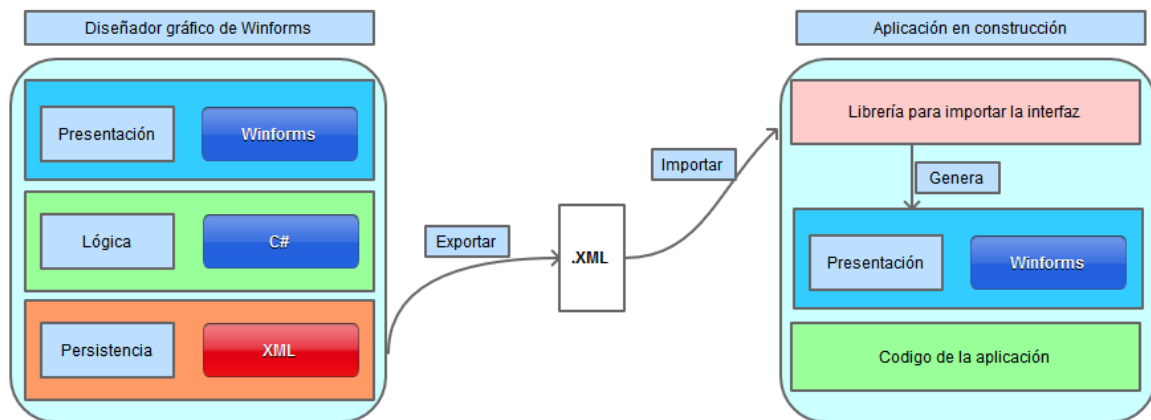


Figura 1. Estructura do sistema xerador de Winforms

A vista da aplicación é desenvolva coa API Winforms e non con WPF ou outra API de interface para que a aplicación poida ser utilizada en sistemas Windows e GNU/Linux sen problemas. O mesmo ocorre coa persistencia que se fará en Xml e non en Xaml, debido a que este último non esta soportado por Mono actualmente.

3.1. Winforms vs Windows Presentation Foundation vs GTK#

Cando se comezou a deseñar o sistema o primeiro que se plantexou foi con qué tecnoloxía se implementará a interface do sistema. Inmediatamente apareceron tres opcións: Winforms, WPF e GTK#.

A principal característica que diferencia estas dúas APIs é a súa relación co resto do código da aplicación. Mentres que en Winforms e GTK# o deseño da interface forma parte da lóxica da aplicación en C#, WPF permite separar completamente a capa de presentación da

lóxica do sistema permitindo definir as interfaces de usuario en ficheiros Xaml. Esta opción permitiría crear un código moito mellor estruturado pero por desgracia Mono non admite este tipo de interface e unha das principais características do sistema é que poderá ser utilizado tanto en Windows .Net como en GNU/Linux Mono polo que queda totalmente descartado.

Nos encontramos ante a decisión de se utilizar GTK# o Winforms para a nosa aplicación. As dúas opcións a nivel de programación son moi similares pero finalmente utilizarase Winforms. A explicación desta decisión baséase simplemente en que xa que o sistema está destinado a deseñar interfaces coa API Winforms, utilizar esta mesma na propia aplicación fará que o deseño e visualización das interfaces en creación sexa mais fiable e ademais moito mais sinxelo de desenvolver, e o non haber unha razón importante para utilizar GTK#, Winforms é sen dúbida a mellor opción.

3.2. XML vs XAML

En canto á persistencia do sistema tamén nos encontramos cunha decisión en canto ás tecnoloxías a utilizar, Xml ou Xaml. Este ultimo foi creado para a especificación das interfaces de usuario en WPF. Xaml é un linguaxe declarativo baseado en Xml e optimizada para describir interfaces de usuario. Xaml representa directamente a creación de instancias de obxectos nun conxunto concreto de tipos de respaldo definidos nos ensamblados. Esta tecnoloxía sería perfecta para representar os elementos da interface creada coa nosa aplicación, pero de novo temos un problema de compatibilidade, Mono non soporta serialización en Xaml polo que nos vemos obrigados a utilizar Xml normal. Pero non todo é malo, en Mono e .Net existen as clases XmlReader e XmlWriter que permiten serializar obxectos de C# en ficheiros Xml permitindo polo tanto gardar instancias deses obxectos facendo a diferenza entre Xml e Xaml se reduce a temas de rendemento.

3.3. NUnit

Para probar as distintas clases que compoñen o sistema desenvolvido utilizouse o framework NUnit e creáronse unha serie de probas de unidade. NUnit foi portado inicialmente de JUnit. Está escrito totalmente en C# y ha sido completamente rediseñado para aproveitar moitas das características dos linguaxes de .NET.

3.4. Git

Git é un software de control de versións creado pensando na eficiencia e confiabilidade do mantemento de versións de aplicacións cando estas teñen un gran número de arquivos de código fonte. Na miña situación pensei que sería útil utilizar un sistema deste tipo xa que a metodoloxía de desenvolvemento que se utilizou, e da que se falará na seguinte sección, é unha metodoloxía baseada en prototipos e por tanto podería ser útil poder volver a unha versión anterior e estable do prototipo en desenvolvemento. Ademais utilizou se GitHub para aloxar o repositorio externamente o que facilitou poder compartilo co titor do proxecto, e o fixo dispoñible para poder traballar nel dende diferentes equipos e sistemas dunha forma cómoda.

4. Metodoloxías utilizadas

Unha metodoloxía de desenvolvemento de software refírese a un framework que é usado para estruturar, planear e controlar o proceso de desenvolvemento nos sistemas de información.

Ó longo do tempo foron surxindo moitos métodos diferentes cada un con puntos fortes e débiles. Cada unha destas metodoloxías teñen o seu propio enfoque para o desenvolvemento de software. Entre os distintos enfoques destacamos catro como os mais xerais.

Primeiro o modelo en cascada, este consiste nun proceso secuencial de desenvolvemento no que os pasos do desenvolvemento son vistos cara abaixo a través das fases de análises das

necesidades, o deseño, implementación, probas, a integración e o mantemento.

Outro enfoque destacado é o prototipado. O prototipado consiste en desenvolver modelos de aplicacións software que permiten visualizar a funcionalidade básica da mesma, sen precisar incluír toda a lóxica ou todas as características do modelo finalizado. O prototipado permite ó cliente avaliar de maneira temprana o produto e interactuar cos deseñadores e desenvolvedores para saber se se está a cumprir coas expectativas e as funcionalidades acordadas.

O enfoque incremental é unha mestura dos dous enfoques anteriores. Este enfoque provee dunha estratexia para controlar a complexidade e os resgos, desenvolvendo unha parte de produto software reservando o resto de aspectos para o futuro. Conta cunha serie de pequenas cascadas, onde tidas as fases da cascada se completan para unha pequena parte dos sistemas antes de proceder co próximo incremento.

Por último o enfoque de desenvolvemento en espiral, nel a atención céntrase na avaliación e redución de resgos do proxecto dividindo o proxecto en segmentos máis pequenos e proporcionar máis fiabilidade de cambio durante o proceso de desenvolvemento, así como ofrecer a oportunidade de avaliar os resgos co peso da consideración da continuación do proxecto durante todo o ciclo de vida. Cada viaxe ó redor da espiral atravesa catro fases, determinar obxectivos, alternativas e desencadéante da iteración; avaliar as alternativas, identificar e resolver resgos; desenvolver e verificar os resultados da iteración, e plan da próxima iteración.

Para desenvolver o noso sistema optaremos pola utilización dun **modelo de desenvolvemento baseado en prototipos**.

O modelo de baseado en prototipos é un modelo de desenvolvemento evolutivo que comeza coa definición dos obxectivos globais para o desenvolvemento do proxecto, posteriormente se identifican os requisitos coñecidos e as áreas do esquema. Tras isto plantéxase con rapidez unha iteración de construción de prototipos e se presenta o modelado. O resultado desta iteración é un prototipo que se presentará ós clientes para que o avalíen e proporcionen información sobre os defectos do prototipo para poderlos corrixir en prototipos futuros.

Con esta metodoloxía lógrase un mellor entendemento dos obxectivos e características da

aplicación así como a posibilidade de refinar pouco a pouco o conxunto da aplicación para ó final obter como resultado o sistema desexado.

Existen dous tipos de prototipos, o prototipo desechable e o prototipo evolutivo. O primeiro é utilizado unicamente para desenvolver unha parte do sistema pouco clara para probala e eliminar dúbidas acerca dela. Tras isto o prototipo se descarta e esa funcionalidade se engade o sistema completo ou a outro prototipo. E o prototipo evolutivo é un modelo parcialmente construído e que é mellorado en cada fase de prototipado engadíndolle mais funcionalidades, obtendo finalmente un prototipo que se corresponde ó sistema final e este prototipo deixa de selo para ser o produto a entregar.

O desenvolvemento coa metodoloxía baseada en prototipos consta das seguintes fases:

- Investigación preliminar: Nela defínese o problema, os obxectivos xerais do sistema e un deseño preliminar e rápido do sistema completo.
- Fase de prototipado: Nela desenvólvense os prototipos que se irán presentando ó cliente e iran refinándose ata obter o sistema final. Esta fase compóñese das seguintes subfases cíclicas:
 - Deseño rápido: Nesta fase defínense as funcionalidades que se van a implementar no prototipo e faise un deseño deste.
 - Construción do prototipo: Implementase o prototipo definido na fase anterior e fanse as probas necesarias.
 - Avaliación: Preséntase o prototipo ó cliente para que sexa avaliado e poder recibir indicacións para mellorar o sistema para á próxima iteración.
- Desenvolvemento do produto: Creación do sistema final produto da evolución do sistema mediante a fase de prototipado.

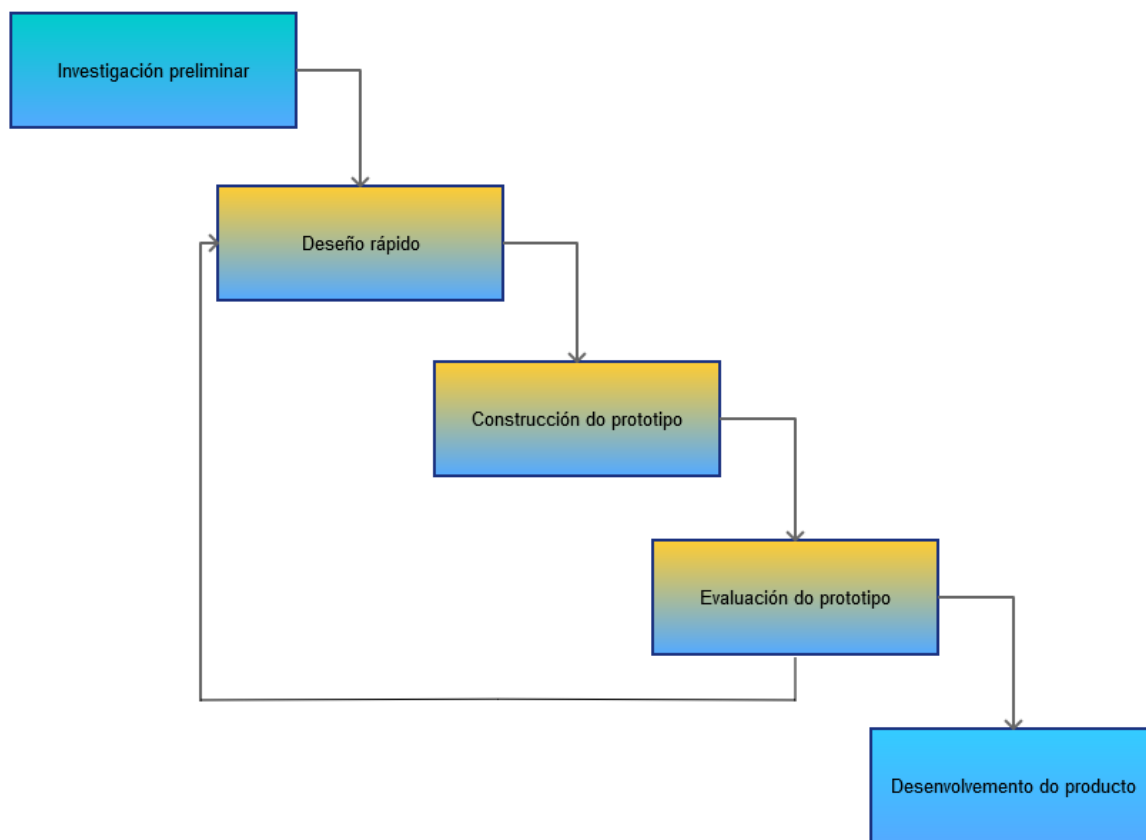


Figura 2. Modelo de Prototipos.

5. Planificación e presuposto

5.1. Planificación

Nesta sección móstrase unha comparativa entre a planificación estimada e a temporización final para este proxecto.

A planificación inicial do proxecto resúmese na Táboa 1. Nesta táboa pódese observar que a duración estimada do proxecto era de 300 horas, repartidas en 10 semana de traballo. Estes cálculos estaban realizados en base a unha dedicación prevista de 5 días á semana, traballando 6 horas ó día. Para realizar a planificación inicial do proxecto decidiuse establecer unha duración de 2 semanas para cada un dos 3 prototipos previstos.

Dedicación semanal prevista (en horas/semanas): 30	
Fase	Estimación temporal (en semanas)
Captura de requisitos xerais do sistema	1
Prototipo 1	2
Prototipo 2	2
Prototipo 3	2
Construción final do sistema	1
Documentación	10
TOTAL PROXECTO	10

Táboa 1. Resumo da planificación estimada

A Figura 3 representa a duración e as datas de planificación estimada. Como pode observarse, a data estimada de comezo foi o 07/04/2014 e a data estimada de finalización era o 20/06/2014.




		Nombre	Duración	Inicio	Fin
1		Captura de requisitos generales del sistema	1s	07/04/2014	11/04/2014
2		▢ Prototipo 1	2s	21/04/2014	02/05/2014
3		Definición de requisitos del prototipo	1d	21/04/2014	21/04/2014
4		Diseño del prototipo	2d	22/04/2014	23/04/2014
5		Construcción del prototipo	1s	24/04/2014	30/04/2014
6		Evaluación del resultado	2d	01/05/2014	02/05/2014
7		▢ Prototipo 2	2s	05/05/2014	16/05/2014
8		Definición de requisitos del prototipo	1d	05/05/2014	05/05/2014
9		Diseño del prototipo	2d	06/05/2014	07/05/2014
10		Construcción del prototipo	1s	08/05/2014	14/05/2014
11		Evaluación del resultado	2d	15/05/2014	16/05/2014
12		▢ Prototipo 3	2s	19/05/2014	30/05/2014
13		Definición de requisitos del prototipo	1d	19/05/2014	19/05/2014
14		Diseño del prototipo	2d	20/05/2014	21/05/2014
15		Construcción del prototipo	1s	22/05/2014	28/05/2014
16		Evaluación del resultado	2d	29/05/2014	30/05/2014
17		Construcción del sistema final	1s	02/06/2014	06/06/2014
18		Documentación	10s	07/04/2014	20/06/2014

Figura 3. Planificación estimada

Por último, na Figura 4 amósase o diagrama de Gantt correspondente á planificación estimada, xunto coas fases do proxecto.

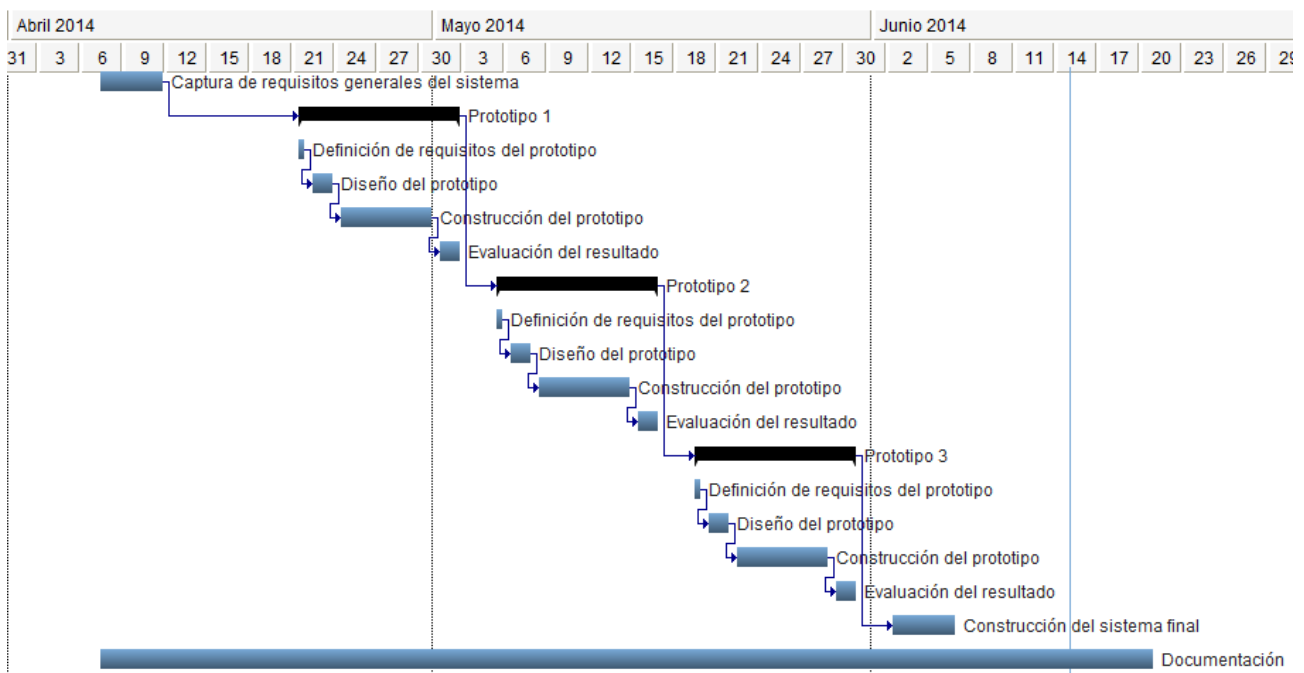


Figura 4. Diagrama de Gantt da planificación estimada

A variación mais importante que sufriu a planificación inicial foi na construción do terceiro prototipo. Este tivo mais traballo que os outros dous prototipos e a súa construción durou unha semana mais, e como consecuencia, aumentou o número de horas dedicadas para o proxecto e o retraso da finalización do mesmo. Esta variación pódese observar na Táboa 2 e na Figura 5 e Figura 6 correspondentes ó diagrama de Gantt resultado da temporización real.

Dedicación semanal prevista (en horas/semanas): 30	
Fase	Estimación temporal (en semanas)
Captura de requisitos xerais do sistema	1
Prototipo 1	2
Prototipo 2	2
Prototipo 3	3
Construción final do sistema	1
Documentación	10,4
TOTAL PROXECTO	10,4

Táboa 2. Resumo da temporización final do proxecto




		Nombre	Duración	Inicio	Fin	Predecesoras
1		Captura de requisitos generales del sistema	1s	07/04/2014	11/04/2014	
2		☐ Prototipo 1	2s	21/04/2014	02/05/2014	1
3		Definición de requisitos del prototipo	1d	21/04/2014	21/04/2014	
4		Diseño del prototipo	2d	22/04/2014	23/04/2014	3
5		Construcción del prototipo	1s	24/04/2014	30/04/2014	4
6		Evaluación del resultado	2d	01/05/2014	02/05/2014	5
7		☐ Prototipo 2	2s	05/05/2014	16/05/2014	2
8		Definición de requisitos del prototipo	1d	05/05/2014	05/05/2014	
9		Diseño del prototipo	2d	06/05/2014	07/05/2014	8
10		Construcción del prototipo	1s	08/05/2014	14/05/2014	9
11		Evaluación del resultado	2d	15/05/2014	16/05/2014	10
12		☐ Prototipo 3	3s	19/05/2014	06/06/2014	7
13		Definición de requisitos del prototipo	1d	19/05/2014	19/05/2014	
14		Diseño del prototipo	2d	20/05/2014	21/05/2014	13
15		Construcción del prototipo	2s	22/05/2014	04/06/2014	14
16		Evaluación del resultado	2d	05/06/2014	06/06/2014	15
17		Construcción del sistema final	1s	09/06/2014	13/06/2014	12
18		Documentación	10.4s	07/04/2014	24/06/2014	

Figura 5. Temporización final do proxecto

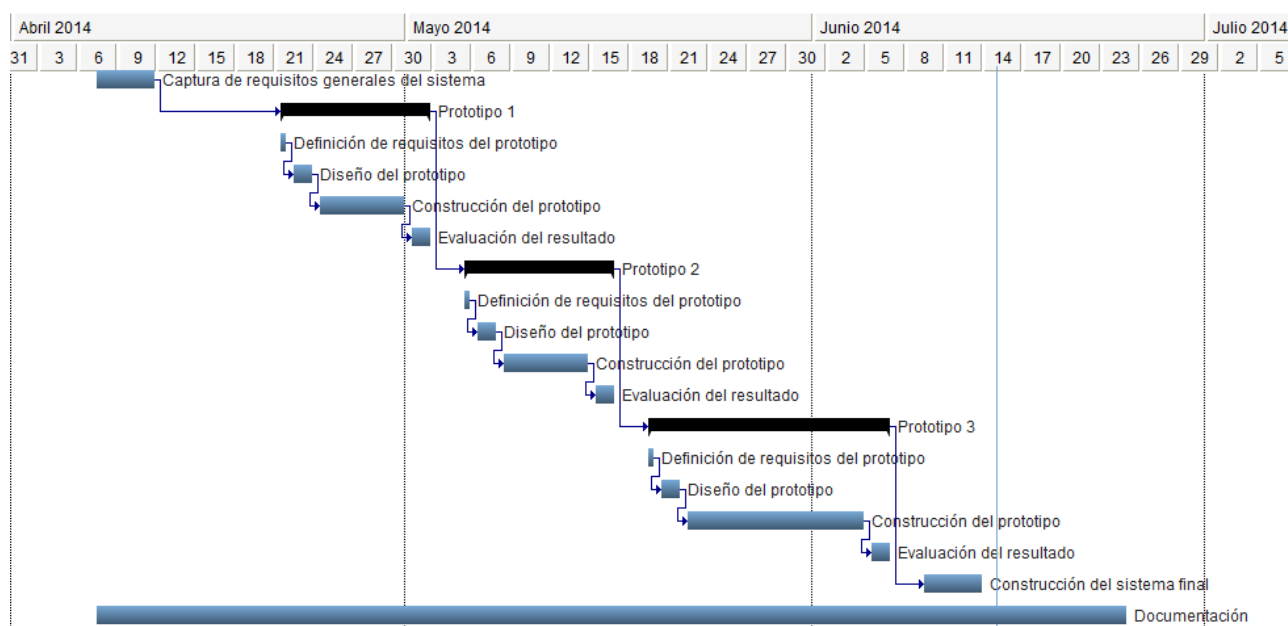


Figura 6. Diagrama de Gantt da temporización real

5.2. Presuposto

En canto ós custos asociados ó hardware, software e recursos humanos, amósase nas seguintes seccións táboas a modo de resumo.

5.2.1 Custo do hardware

Na Táboa 3 detállase o hardware empregado para levar a cabo este proxecto e o seu custo.

Compoñente	Modelo	Custo
Portátil	Hp pavilion dv6 7003 ss	900 €

Táboa 3. Custo do hardware

Para calcular o custo real dos materiais utilizados, xa que o hardware descrito non vai estar dedicado unicamente para este proxecto, fixéronse ás seguintes consideracións;

- Considerándose un uso posible do portátil de 5 días á semana durante ós 12 meses do ano e cunha xornada de 8 horas o que fai un numero de 2080 horas ó ano.
- Fíxase o tempo de vida útil do portátil en 4 anos, e dicir, 8320 horas.

	Vida útil	Custo total	Custo por horas	Horas de uso	Total amortizado
Portátil	4 anos (8320 horas)	900,00 €	0,109 €	312	34,01 €

Táboa 4. Amortización do proxecto

5.2.2. Custo do software

Na Táboa 5 indicase o custo do software empregado.

Software	Custo
Linux mint 16 Petra	0 €
Monodevelop	0 €
Xamarin Studio	0 €
GitHub	0 €
Pencil	0 €
Ganttter.com	0 €
Visal Paradigm (Licenza aportada pola Universidade de Vigo)	0 €
Libre Office	0 €
Total	0 €

Táboa 5. Custo do software empregado

Debido a que todo o software é de licenza libre ou gratuíta, o custo total foi de 0 € e non é preciso indicar á amortización do mesmo.

5.2.3 Custo do persoal do proxecto

Para o cálculo do custo do persoal considerouse un prezo da hora de traballo de 20€ para unha duración total do proxecto de 312 horas.

Perfil	Horas	Prezo/Hora	Custo total
Analista/Programador	312 h	20 €/h	6240 €

Táboa 6. Custo do persoal

5.2.4 Custo total do proxecto

Tendo en conta os custo de hardware, software e de persoal, na Táboa 7 indícase o cálculo total dos custos do proxecto.

Concepto	Custo
Hardware	34,01 €
Software	0 €
Persoal	6240 €
TOTAL	6274,01 €

Táboa 7. Custos totais do proxecto

6. Problemas atopados e solucións aportadas

Un dos principais problemas atopados durante o desenvolvemento do sistema tivo orixe no

obxectivo de facer un sistema multiplataforma e sinxelo. Aínda que Mono é a alternativa libre e para sistemas GNU/Linux de .Net existen moitas diferenzas de comportamento. Houbo obxectos da interface que nun sistema funcionaban dunha forma e no outro doutra, como por exemplo a hora de rechear un DataGridView, en windows tan só permite rechear as celdas con Strings e en Linux dá a posibilidades de utilizar obxectos. Finalmente descubrín que o mais restritivo é windows e facéndoo de forma que funcione ben aí en Linux con Mono funciona correctamente tamén.

Con respecto ó mesmo tema houbo outro problema que se debe mencionar. Inicialmente a persistencia da ferramenta e a exportación das interfaces creadas con ela quería facerse co formato Xaml pero debido a que este non está soportado por Mono finalmente houbo que decidir utilizar serialización con Xml.

Outro inconveniente sufrido foi ao principio da implementación decidir como crear a previsualización das interfaces en deseño. Non se sabía se se deberían utilizar imaxes ou outro sistema. Finalmente o mais sinxelo, práctico e efectivo foi renderizar directamente a interface coma se fora parte da interface do sistema.

7. Futuras ampliacións

Neste proxecto centrámonos na posibilidade de poder crear interfaces gráficas de usuario sinxelas, cos elementos máis coñecidos e utilizados da API Winforms, pero aínda quedan controles que non se incluíron e que, nun futuro, podería ser interesante incluír ata o punto de dar soporte a todos os elementos da API. Ademais da cantidade de elementos da API que non se soportan, nos elementos que si se incluíron non é posible personalizalos tanto coma se podería facer a través de código sen utilizar a aplicación, existen tamén moitas propiedades que non se lles dá a posibilidade de modificar. O mesmo ocorre cos eventos, cada elemento da API Winforms ten unha gran cantidade de eventos dos cales só os máis importantes e comúns están habilitados para a súa edición a través da ferramenta.

Outra ampliación que se podería contemplar é dar a posibilidade de crear e editar múltiples

formularios de forma simultánea e así poder rapidamente e de forma cómoda construír a totalidade das interfaces da aplicación na que as quereremos importar.

8. Bibliografía

Referencias Web

- [1] - <http://msdn.microsoft.com/>. Documentación das librerías de C#
- [2] - <http://www.nunit.org/>. Documentación do framework de probas NUnit.
- [3] - <http://git-scm.com/>. Documentación acerca do sistema de control de versións Git.
- [4] - <http://mono-project.com/>. Documentación acerca do framework Mono.

Libros

- [1] - Windows Forms Programming in C# (Microsoft .Net Development Series).
Chris Sells
- [2] - .NET Windows Forms in a Nutshell
Ian Griffiths , Matthew Adams
- [3] - C# 5.0 in a Nutshell: The Definitive Reference
Joseph Albahari, Ben Alahari

MANUAL DE TÉCNICO

1. Introducción

Este documento farase un recorrido por toda a fase de desenvolvemento do proxecto, aportando os diagramas de deseño do sistema, co fin de facilitar as labores de mantemento e ampliación do sistema nun futuro.

A metodoloxía de desenvolvemento que se utilizou foi unha metodoloxía baseada en prototipos. Esta metodoloxía baséase na construción de prototipos, isto é, a construción dunha aplicación que contén un subconxunto dos obxectivos e funcionalidades do sistema. Mediante a construción dun prototipo evolutivo estímase que coa finalización do terceiro prototipo este sexa o sistema final completo.

Nesta metodoloxía primeiramente faise unha análise preliminar dos obxectivos do sistema e determínase a grandes rasgos a estrutura que terá. Logo pasarase a creación dos prototipos para finalmente entrar na fase de desenvolvemento do produto final que consiste nada máis ca converter o último prototipo desenvolvido no sistema final.

A creación de cada prototipo divídese en tres fases. Unha fase de deseño rápido onde se observará o deseño da estrutura do sistema a través dun diagrama de clases, e o conxunto de funcionalidades mediante un diagrama de casos de uso. Ademais para comprender mellor o funcionamento interno dos casos de uso aportaranse diagramas de secuencia de sistema. A seguinte fase é a fase de construción onde se implementará o prototipo seguindo o deseño aportado e finalmente se farán unhas pequenas probas de unidade. E por último a dáse de avaliación e retroalimentación na cal nos reuniremos co titor do proxecto para facer unha demostración de funcionamento do sistema e determinar os erros que se deberán corrixir no seguinte prototipo e as novas funcionalidades que debe ter.

A utilización desta metodoloxía de desenvolvemento permítenos crear rapidamente o sistema e conseguir un sistema moi probado debido a que se publicaron diversos prototipos e cada un deles foi sometido a unha pequena fase de proba real. Pero como contrapartida o nivel de documentación é mínimo, redúcese unicamente ós diagramas utilizados para o deseño rápido de cada un dos prototipos.

2. Investigación preliminar

O proxecto que vaise a desenvolver trátase dun sinxelo sistema que permita a creación de interfaces gráficas ca API Winforms de C# para mais tarde poder importalas nun proxecto en desenvolvemento axeo a este sistema.

O sistema comporase da ferramenta de deseño e dunha librería para poder utilizar as interfaces creadas, noutro proxecto en desenvolvemento.

A ferramenta debe permitir, non todos os controles existentes na API, pero si os máis comúns e utilizados. Da mesma forma o nivel de personalización destes controles debe centrarse nas propiedades mais importantes.

Tras analizar os distintos controles da API Winforms púidose xerar un diagrama de clases (Figura 7) moi sinxelo de modo que nos poida servir para ter unha idea inicial de como será o sistema, e así, a partir del deseñar a estrutura dos seguintes prototipos en base ás funcionalidades e requisitos que cubrirán.

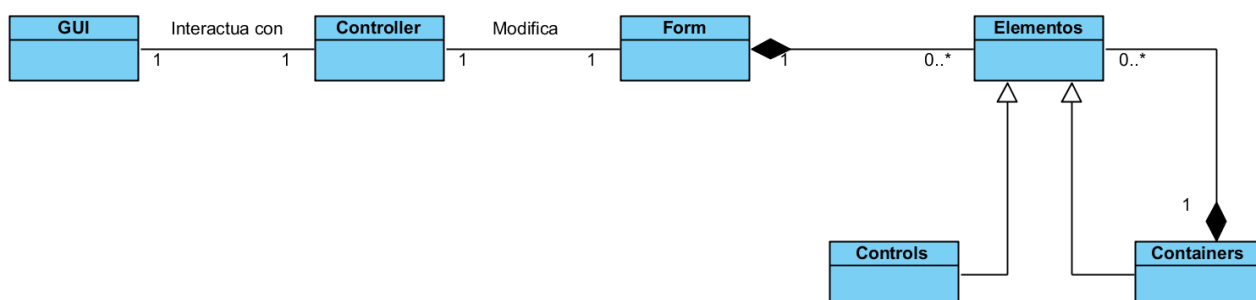


Figura 7. Diagrama de clases

GUI: Interface gráfica da ferramenta.

Controller: Clase intermediaria entre a interface e o formulario.

Form: Clase que representa o formulario que se está a deseñar coa ferramenta.

Elementos: Representa ó conxunto de obxectos que se lle poden engadir á interface.

Controls: Correspóndese ós elementos simples da API Winforms.

Containers: Elementos da API Winforms que serven para conter outros elementos.

Con respecto ó deseño da librería, o diagrama de clases correspondente é o mesmo ó da ferramenta pero eliminando a clase "GUI".

Para poder comprender mellor as funcionalidades principais que se esperan da ferramenta de deseño, e nos seguintes prototipos poder repartilas entre os prototipos planificados, onde se estenderán mais, creouse un pequeno diagrama de casos de uso (Figura 8).

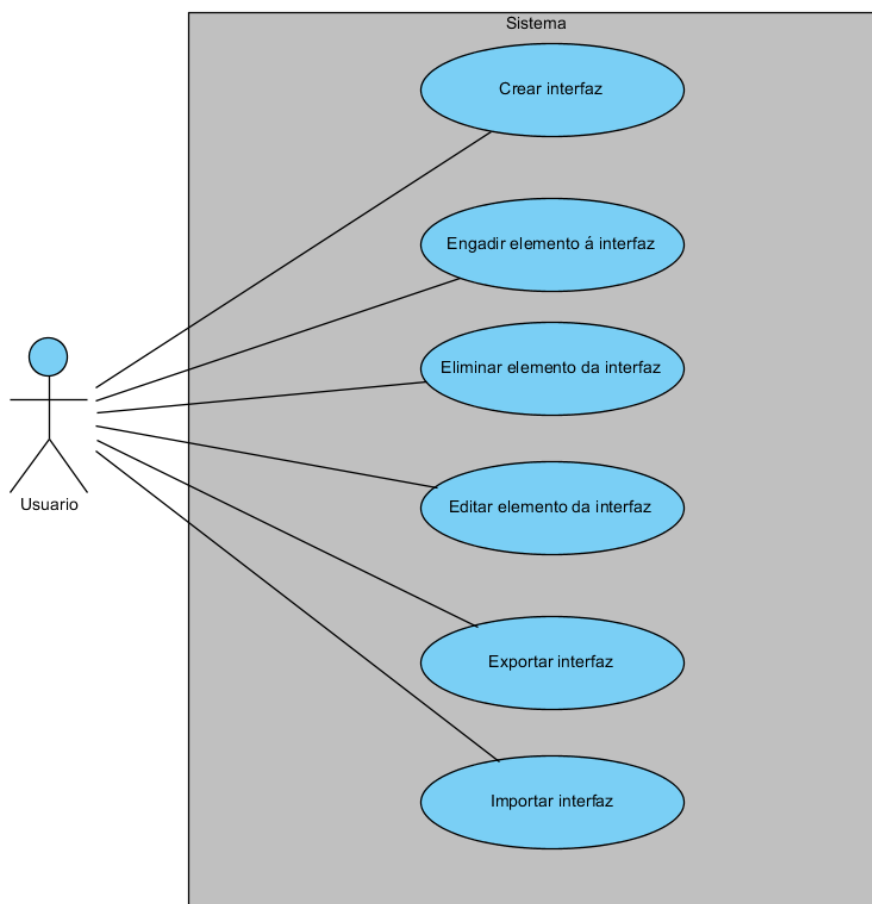


Figura 8. Diagrama de casos de uso

Crear Interface

Este caso de uso indica que o sistema deber permitir a creación dunha nova interface gráfica para poder deseñala na ferramenta.

Engadir elemento á interface

O sistema debe permitir engadir múltiples elementos á interface en deseño.

Editar elemento

Debe ser posible editar as propiedades correspondentes ós elementos que compoñen a interface en deseño.

Eliminar elemento da interface

Debe existir a posibilidade de eliminar elementos presentes na interface que se está a deseñar.

Exportar interface

Esta funcionalidade do sistema fai referencia a que se debe permitir exportar a interface que se ha deseñado coa ferramenta para así dispoñer dela noutro proxecto.

Importar interface

Debe ser posible importar a interface que anteriormente foi exportada para poder continuar co seu deseño o modificala.

No caso da biblioteca, tan só existe un caso de uso, este é "Importar interface". Da mesma forma que se importa a interface para podela modificar, tamén se pode importar nun proxecto C# en desenvolvemento.

Para ter unha idea un pouco mais concreta de cal será o funcionamento interno do sistema para cada un dos casos de uso especificados, desenvolvéronse os correspondentes diagramas de secuencia os cales pódense apreciar nas Figuras 9,10,11,12,13 e 14.

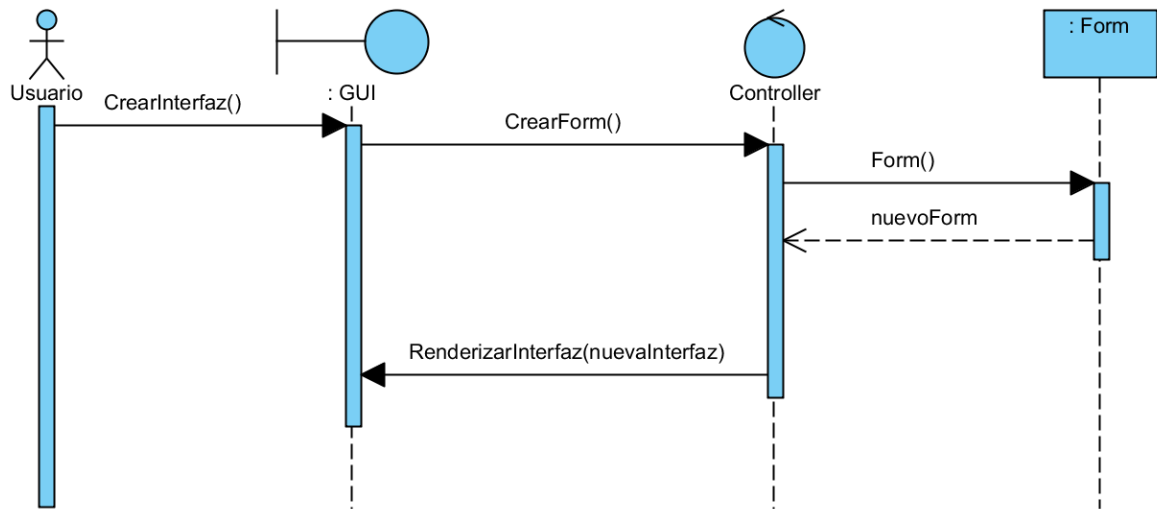


Figura 9. Diagrama de secuencia. Crear interface.

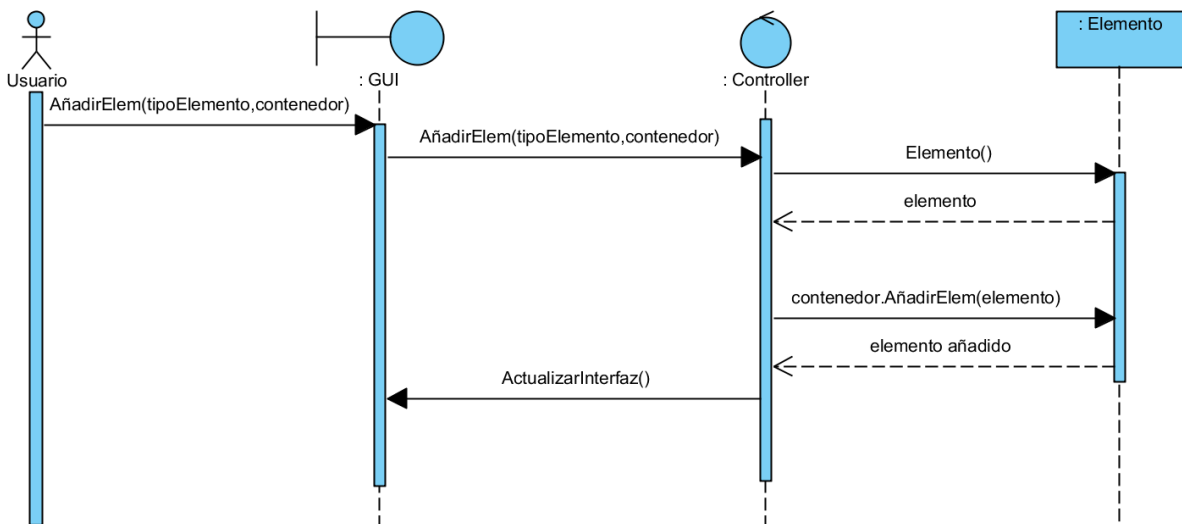


Figura 10. Diagrama de secuencia. Engadir elemento.

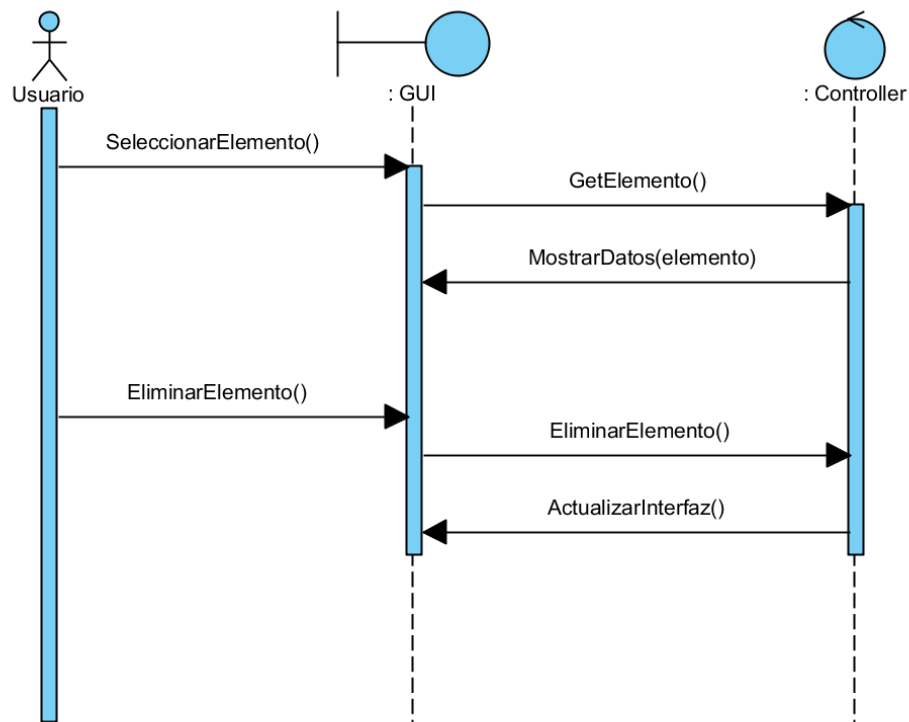


Figura 11. Diagrama de secuencia. Eliminar elemento.

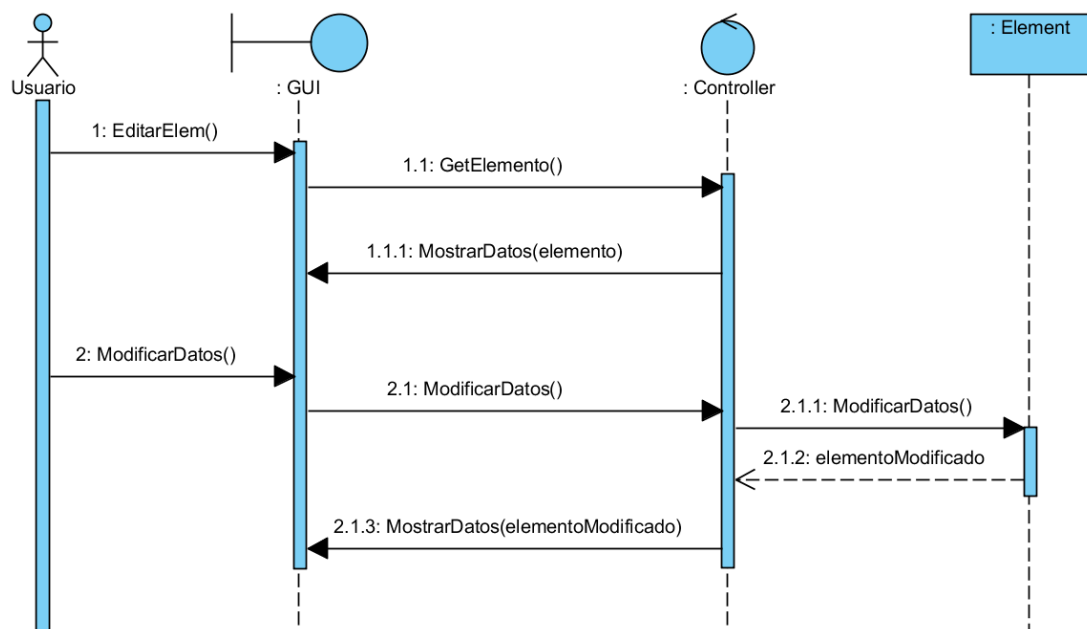


Figura 12. Diagrama de secuencia. Editar elemento.

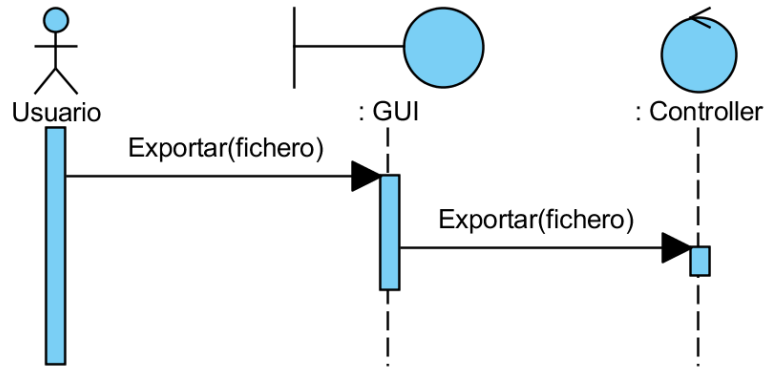


Figura 13. Diagrama de secuencia. Exportar interface.

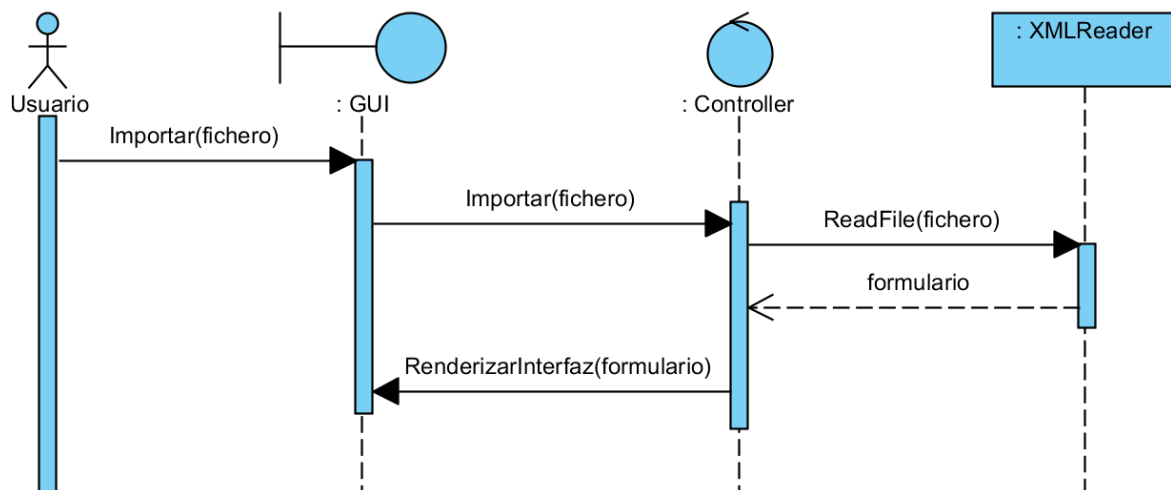


Figura 14. Diagrama de secuencia. Importar interface.

Tras esta primeira aproximación a la arquitectura y el funcionamiento del sistema xa

podemos comezar co desenvolvemento dos prototipos.

3. Primeiro prototipo

3.1. Deseño rápido

Para o primeiro prototipo tívose como obxectivo a creación dunha ferramenta de deseño sinxela. Esta ferramenta dará soporte á creación da interface e poderlle engadir uns poucos elementos. Estes elementos son:

- Containers
 - Border
 - HBox
 - VBox
 - Grid
- Controles
 - Button
 - Label
 - TextBox

Este é o conxunto de controles que se pensou que era o mais sinxelo deixando os mais complicados para os seguintes prototipos xa que neste primeiro prototipo tense que construír a interface gráfica completa.

Para deseñar a interface, fíxose o boceto da Figura 15, no cal pódese observar que a interface se compondrá dun treeview no lateral esquerdo, no cal se mostrarán os diferentes elementos engadidos na interface e desenvolvemento. Na parte dereita pódese ver unha táboa na cal se poderán visualizar i editar as propiedades dun dos elementos engadidos á interface. Finalmente cóntase cunha zona central, a zona de traballo, onde se poderá ir visualizando a interface que se está a deseñar.

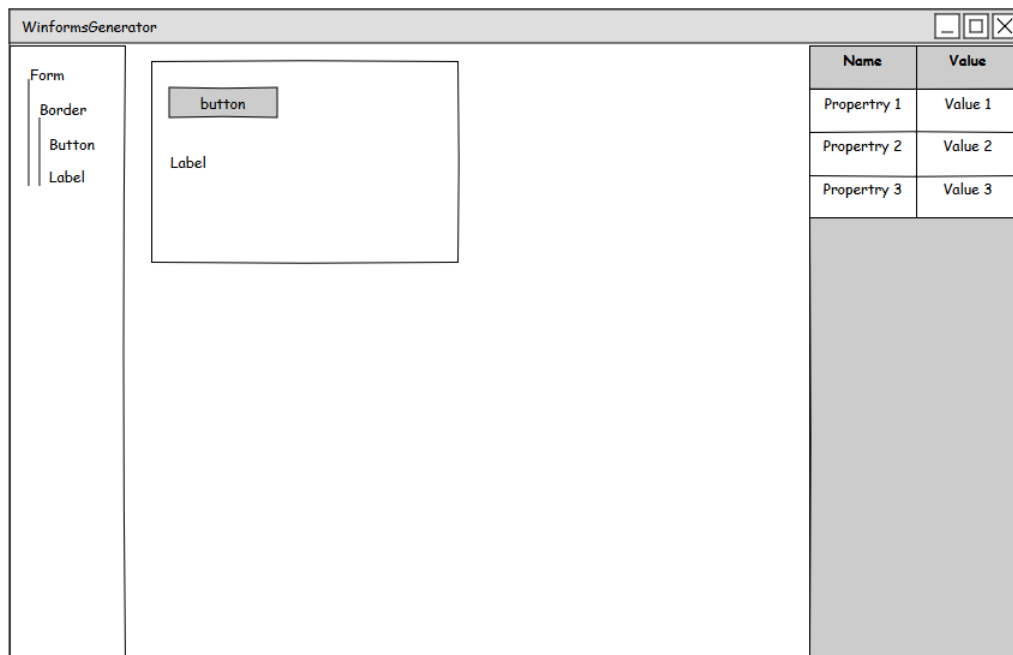


Figura 16. Boceto da interface de usuario

Tendo en conta estas funcionalidades principais coas que contará o primeiro prototipo, na Figura 17 pódese observar o diagrama de casos de uso equivalente.

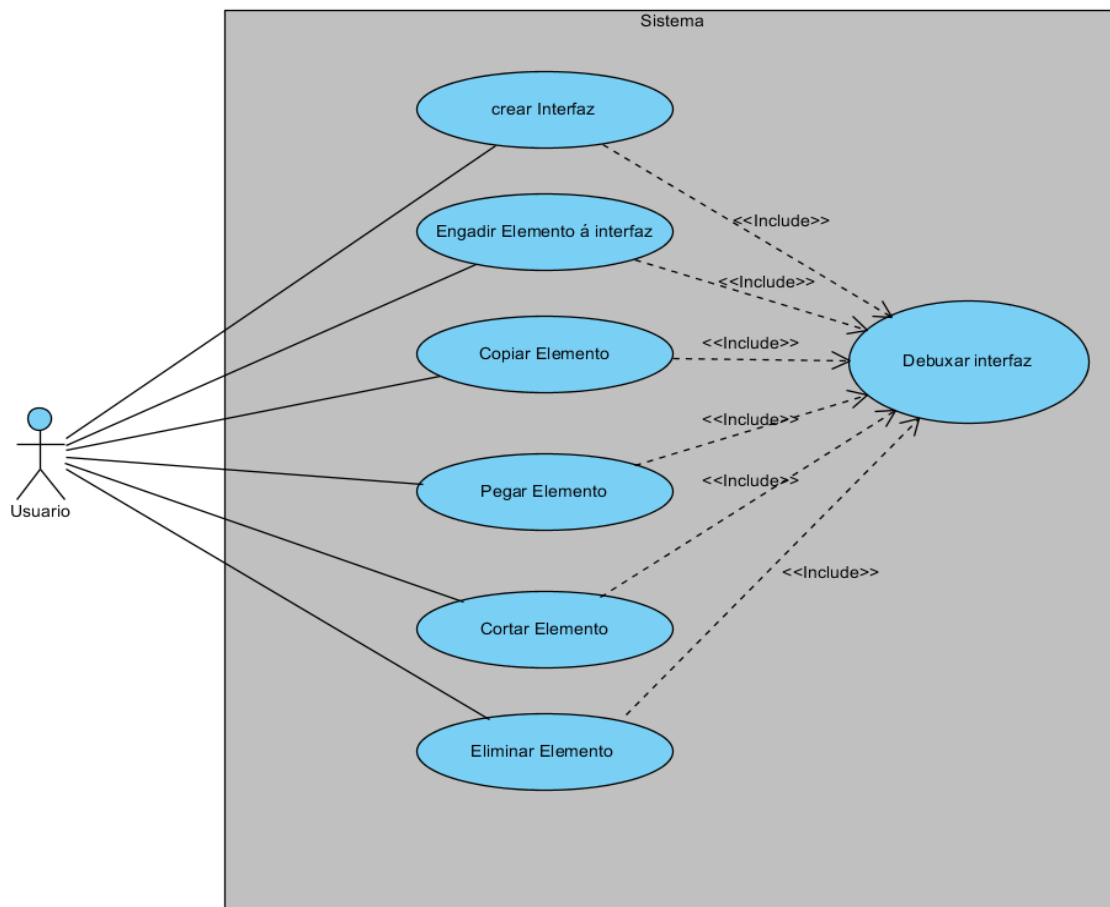


Figura 17. Diagrama de casos de uso do Primeiro Prototipo

Crear interface

Cada vez que se inicia a ferramenta de deseño de interfaces, créase unha nova interface en branco.

Engadir elemento á interface

Para executar esta acción débese executar dende a interface a acción de engadir nun elemento concreto da interface e seleccionar o elemento que se quere engadir.

Tras esto créase un novo elemento do tipo indicado e engádese ó elemento seleccionado, un *Container* ou o propio *Form*.

Copiar elementos

Primeiramente débese seleccionar o elemento a copiar e logo executar a acción de copiar. Mediante esta acción cópiase tanto o elemento coma os elementos que este contén se se trata dun *Container*.

Pegar Elemento

Tras realizar unha acción de copiado habilitase a posibilidade de pegar o elemento gardado. Tras seleccionar o elemento, un *Container* ou o *Form*, no cal se quere pegar, engádese a ese contenedor o elemento e subelementos que se copiaran.

Eliminar Elemento

Primeiramente débese seleccionar un elemento da interface que queremos eliminar. Tras esto seleccionase a acción de eliminar elemento e este desaparece da interface en desenvolvemento

Cortar Elemento

Simplemente trátase de executar a acción de copiado e a de borrado. Debese seleccionar o elemento que queremos cortar e este cópiase e se elimina da interface en desenvolvemento. Tendo en conta as funcionalidades indicadas no diagrama, o boceto da interface e o diagrama de clases da investigación preliminar, desenvolveuse o diagrama de clases mais detallado da Figura 18.

Debuxar Interface

Tras realizar calquera de estas acción anteriores, excepto o copiado, actualízase toda a interface da aplicación. Esta actualización consiste en redebuxar o panel esquerdo que conten o *TreeView* dos elementos que compoñen a interface que se esta a deseñar, e en redebuxar a vista previa desa interface no panel de traballo central.

Apartir de esta definición de casos de uso desenvolveuse os diagramas de secuencia de sistema que se aportan a continuación. Neles indicase dunha forma un pouco abreviada o funcionamento interno real do sistema para cada unha das funcionalidades.

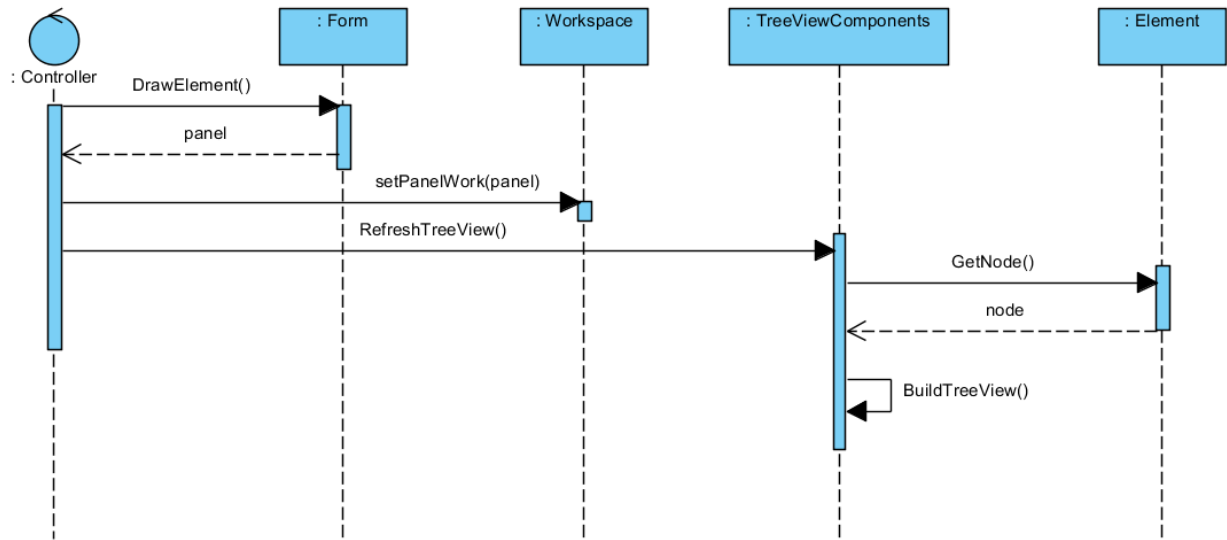


Figura 18. Diagrama de secuencia. Debuxar interface

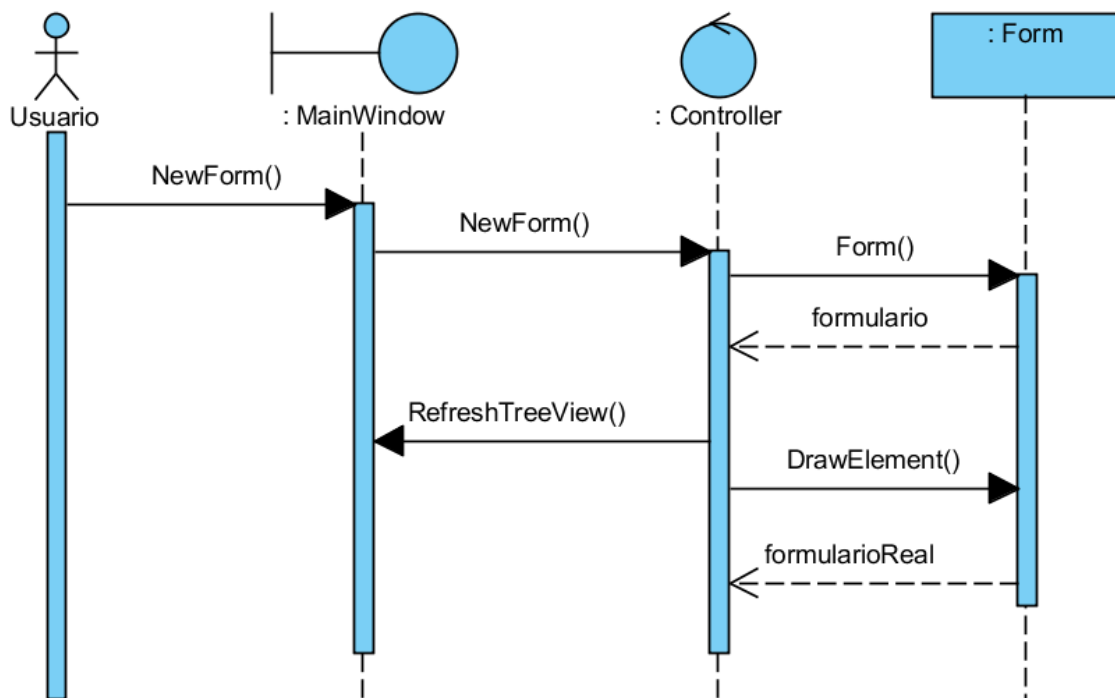


Figura 19. Diagrama de secuencia. Crear interface

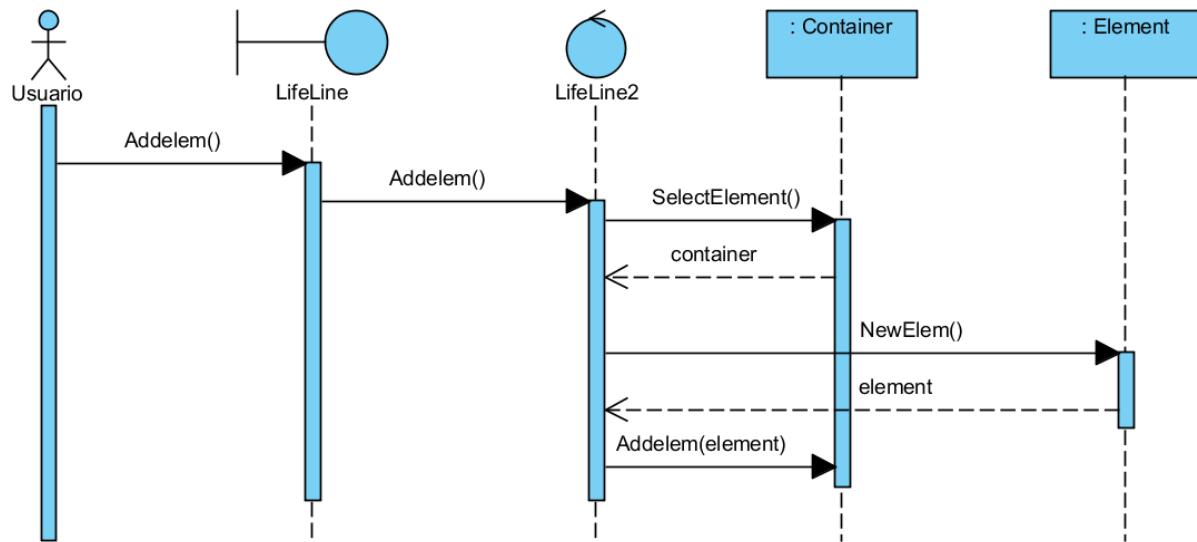


Figura 20. Diagrama de secuencia. Engadir elemento

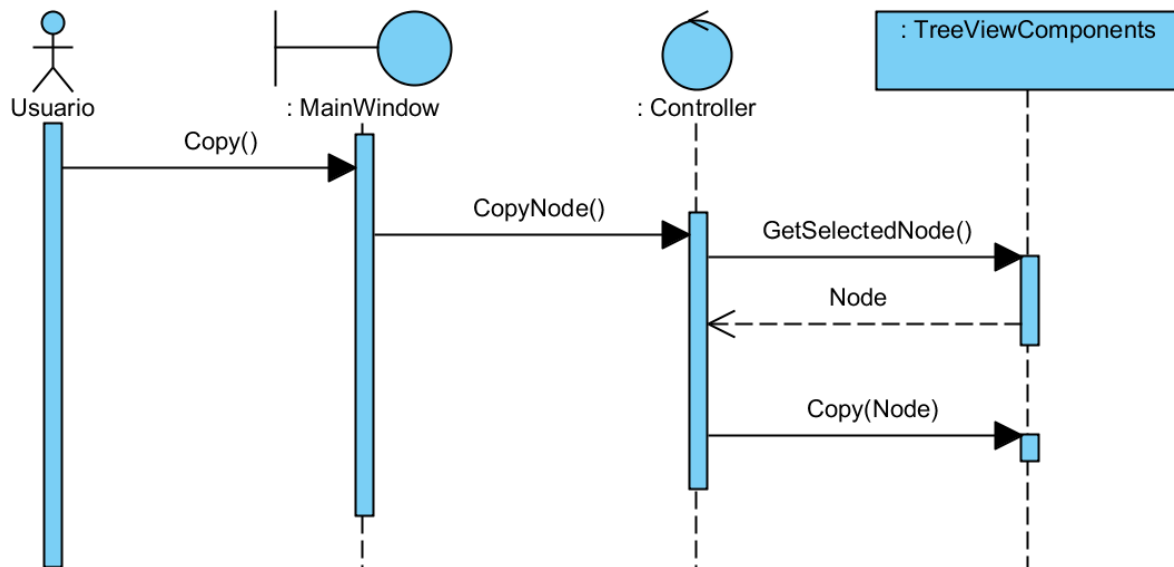


Figura 21. Diagrama de secuencia. Copiar elemento

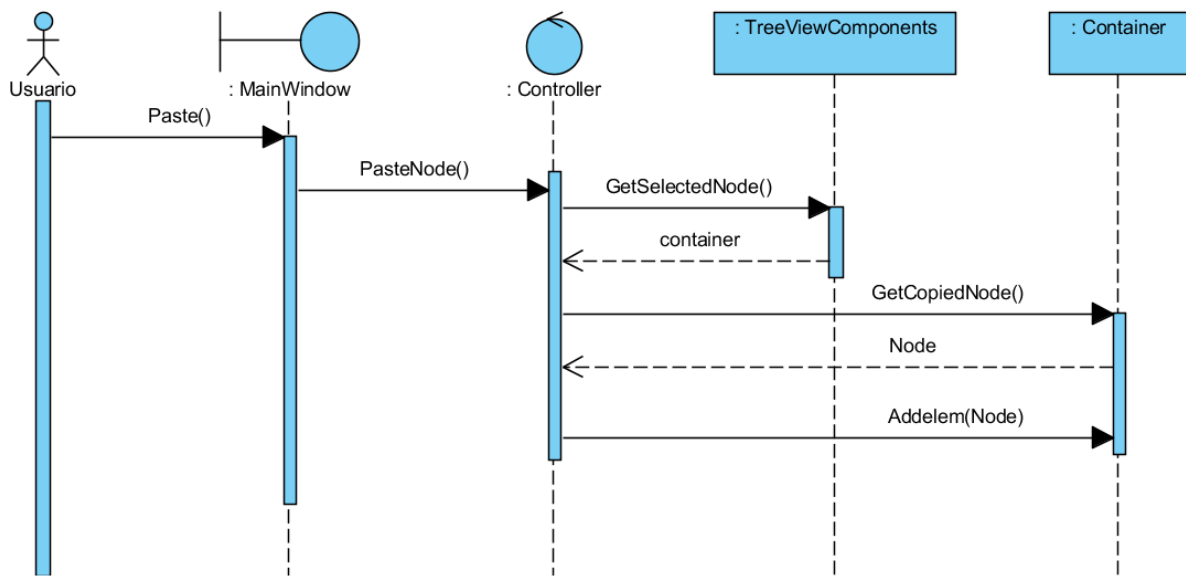


Figura 22. Diagrama de secuencia. Pegar elemento

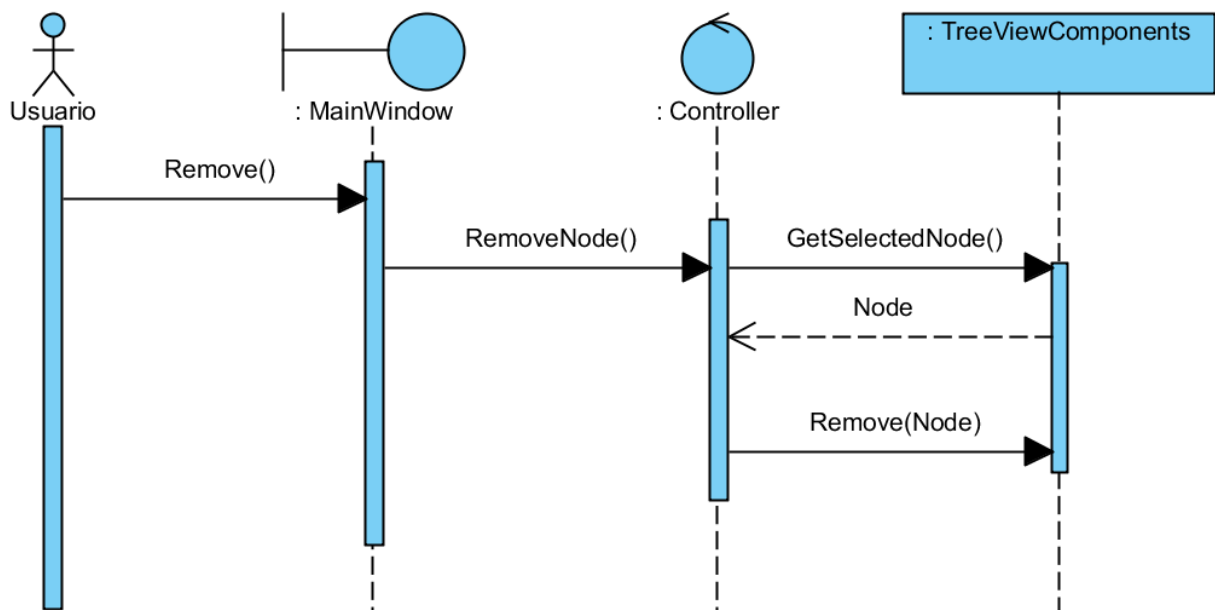


Figura 23. Diagrama de secuencia. Eliminar Elemento

Tendo en conta a especificación do funcionamento do sistema mediante os diagramas de secuencia, xerouse o diagrama de clases da Figura 24, no cal se detalla moito mais a arquitectura do sistema aportando atributos e funcións que deben ter cada unha das clases.

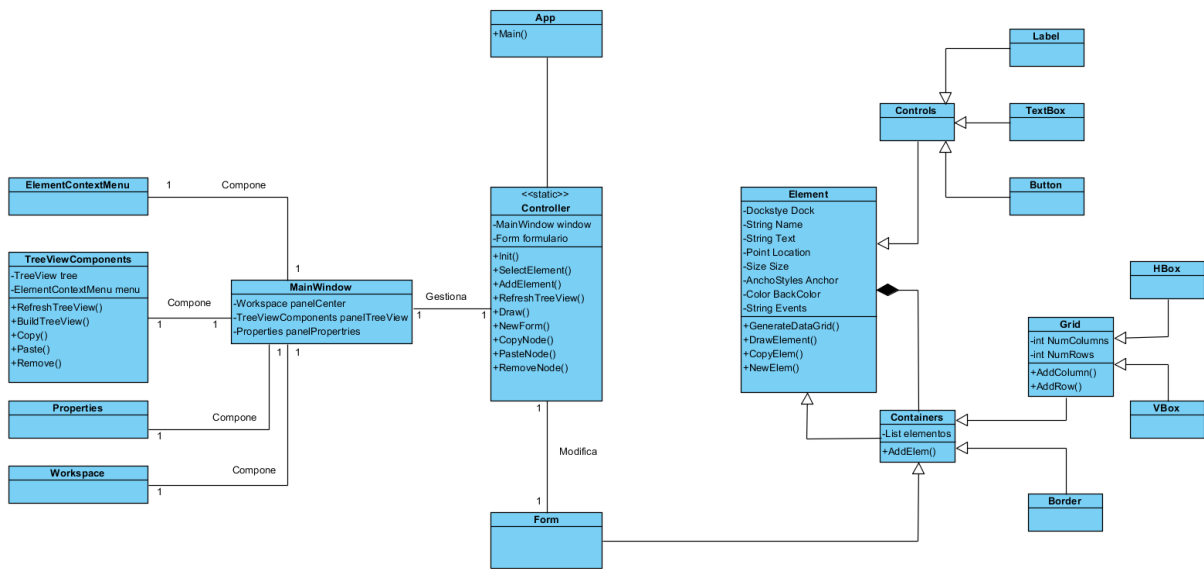


Figura 24. Diagrama de clases do Primeiro Prototipo

App

Esta clase trátase da clase principal do sistema onde se instancia toda a aplicación.

TreeViewComponents

Esta clase trátase dun panel da API Winforms que contén un TreeView, no cal se poderá observar a estrutura da interface que se está a desenvolver coa ferramenta

- **Atributos:**
 - *Treeview tree*: *TreeView* que esquematiza os elementos que forman parte da interface en desenvolvemento
 - *ElementContextMenu menu*: *ContextMenuStrip* o cal se accede co click dereito do rato e que contén as accións que se poder realizar sobre os elementos da interface en desenvolvemento.
- **Métodos:**
 - *BuildTreeView*: Este método serve para debuxar o TreeView a partir da lista de elementos que compoñen a interface.
 - *RefreshTreeView* : Método que serve para actualizar o TreeView e se lle chama tras realizar algunha acción que modifique a estrutura da interface, accións tales como engadir novos elementos ou borrarlos.

- *Copy*: Este método permite copiar unha sección do *TreeView* para pegala mais tarde noutra parte e facendo que se recolquen na interface os elementos representados por esa sección de *TreeView*.
- *Paste*: Método para pegar unha sección de *TreeView* copiada previamente.
- *Remove*: Con este método faise posible eliminar seccións do *TreeView* facendo que os elementos representados por esa sección tamén se eliminen da interface en desenvolvemento.

ElementContextMenu

Esta clase xera un *ContextMenuStrip* que conterá as accións que se poderán executar no sistema, tales como engadir novo elemento, copiar, cortar, pegar e borrar elementos, etc.

Properties

Representa o panel dereito da interface, onde existe un *DataGridView* que conterá as propiedades que se poderán modificar, dun elemento da interface seleccionado previamente.

Workspace

Esta clase contén o panel central da aplicación onde estará a visualización da interface que se está a crear coa ferramenta.

MainWindow

Trátase da ventá principal e única da ferramenta. Está formada por as tres seccións que se puideron apreciar na Figura 16.

- **Atributos:**
 - *Workspace panelCenter*: Panel central da ferramenta onde se poderá observar a construción da interface en desenvolvemento.
 - *TreeViewComponents panelTreeView*: Panel esquerdo da interface onde se poderá atopar o *TreeView* que representa a estrutura da interface en desenvolvemento.
 - *Properties panelProperties*: Panel dereito onde se mostrarán as propiedades a modificar dun elemento da interface.

Element

Esta clase representa ós elementos que se poderán engadir a unha interface en creación.

- **Atributos:**
 - Todas as propiedades desta clase correspóndense coas mesmas propiedades presentes nos controles da API Winforms.
- **Métodos:**
 - *GenerateDataGrid*: Xera a táboa cas propiedades do elemento seleccionado.
 - *DrawElement*: Xera o control da API Winforms equivalente e cos valores das propiedades indicadas na táboa de propiedades do elemento.
 - *CopyElem*: Copia o elemento seleccionado.
 - *NewElem*: Crea un novo elemento.

Controls

Tipo de *Element* que se corresponde cos elementos mais sinxelos que se poderán incluír nunha interface. Del entenden **Label**, **TextBox** e **Button**.

Containers

Tipos de *Elements* que conteñen a outros obxectos da clase *Element*.

- **Atributos:**
 - *List<Elements> elementos*: trátase da lista de elementos que son contidos no propio obxecto.
- **Métodos:**
 - *AddElem*: Método que engade elementos ó *Container*.

Desta clase estenden os elementos **Border** e **Grid**.

Border

Esta clase representa o control *Panel* da API Winforms.

Grid

Esta clase representa ó *TableLayoutPanel* da API Winforms.

- **Atributos:**
 - *int NumColumns*: Número de columnas que ten o *Grid*.
 - *int NumRows*: Número de filas que ten o *Grid*.
- **Métodos:**
 - *AddColumns*: Engade unha columna ó *Grid*.

- *AddRow*: Engade unha fila ó *Grid*.

Desta clase estenden os elementos **Hbox** e **Vbox** que son tipos especiais de *Grid* pero cunha soa fila ou unha soa columna respectivamente.

Form

Esta clase representa o formulario da interface que se vai a construír coa aplicación. Estende de *Container* xa que realmente se trata dese tipo de obxecto aínda que é un pouco especial xa que unicamente existe unha instancia del o mesmo tempo.

Controller

Esta clase trátase dunha clase intermediaria entre á aplicación, a interface gráfica e o formulario da interface en desenvolvemento.

- **Atributos**
 - *MainWindow window*: Instancia da interface gráfica da aplicación.
 - *Form formulario*: Instancia do formulario da interface en desenvolvemento.
- **Métodos**
 - *Init*: Método que inicializa ó controlador creando o formulario para desenvolver a interface e instanciando a interface gráfica da ferramenta.
 - *SelectElement*: Selecciona un elemento da interface en desenvolvemento xerando a táboa de propiedades para poder editala.
 - *AddElement*: Engade un elemento ó formulario e as conseguíntes accións na interface da aplicación.
 - *RefreshTreeView*: Actualiza o *TreeView* da aplicación.
 - *Draw*: Debuxa a visualización da interface que se esta a desenvolver.
 - *NewForm*: Crea un novo formulario.
 - *CopyNode*: Copia unha sección do *TreeView*
 - *PasteNode*: Pega unha sección do *TreeView*, copiada previamente, nunha posición.
 - *RemoveNode*: Elimina unha sección do *TreeView*.

Tras este sinxelo proceso de deseño da interface e da arquitectura e tras a definición das funcionalidades que terá este primeiro prototipo xa podemos avanzar a fase de construción.

3.2. Construción do prototipo

Nesta fase levouse a cabo a implementación do primeiro prototipo. Débese destacar que o mais prioritario á hora de construír o prototipo, sempre foi que permitira, de cara os seguintes prototipos, engadir novos elementos e propiedades da forma mais sinxela posible e co menor número de modificacións no código existente. Isto xustifica a creación da clase *Element* como unha clase abstracta na que se especifican unha serie de funcións que as clases que herdan dela deben implementar. Estes métodos son *DrawElement*, *GetTreeNode*, *CopyElem* e *NewElem*. Ademais proporciona o método virtual *GenerateDataGrid* para que as clases que herdan dela non teñan a necesidade de codificar o método completo tendo xa as partes en común implementadas.

Desta mesma forma as clases *Control* e *Container* que son unhas clases abstractas que herdan de *Element* e que implementan xa para todas as clases que herdan delas o método *GetTreeNode* e deixan como abstractos os métodos *GenerateDataGrid*, *DrawElement*, *NewElem* e *CopyElem*.

Polo tanto os novos elementos que se poidan ir engadindo xa teñen unha parte da lóxica implementada, o que fará que esta labor sexa moito mais sinxela.

Tamén hai que destacar o uso da librería de C# *Reflection* que fai posible a creación de métodos dinámicos, e dicir, unha serie de métodos que dependen dunha clase en concreto, o modificar esta clase cando os métodos se executan o fan de forma diferente, facendo desta forma que ó modificar a lóxica dunha clase non sexa preciso modificar a lóxica destes métodos. Un exemplo é o menú contextual do *TreeView* lateral que representa ó árbore de elementos da interface. Este xérase en función das clases que herdan de *Control* e das que herdan de *Container*. Así o engadir unha nova clase herdeira non é necesaria engadila ó menú xa que este a engade en tempo de execución.

Tras a construción do prototipo desenvólvense unha serie de probas de unidade co fin de facilitar futuras ampliacións ou modificacións do prototipo. Estas probas, desenvoltas con NUnit consisten na proba dos métodos das clases que estenden de *Element* para así comprobar que os Controles da API Winforms que se xeran coa ferramenta son como se

desexa que sexan.

3.3. Avaliación e retroalimentación do prototipo

Nesta fase realizouse la reunión con el titor a modo de cliente, para facer unha demostración do funcionamento do prototipo e determinar qué aspectos do prototipo deben cambiarse ou ampliarse, así como determinar as funcionalidades que deberanse incluír no seguinte prototipo do sistema.

Como puntos de mellora do prototipo decidiuse incluír na interface un menú con botóns que representen as accións de crear novo formulario, copiar, cortar e pegar elemento.

Para ampliar o sistema no seguinte prototipo decidiuse engadir as opcións de gardado e carga da interface en desenvolvemento xa que ata agora non se podía facer. Tamén pensouse en dar opción a probar a interface que se está a crear. Para isto pode que sexa útil comezar xa coa biblioteca de importación da interface noutro proxecto.

4. Segundo prototipo

4.1. Deseño rápido

Neste segundo prototipo o deseño inicial será mais sinxelo xa que tan só consiste en engadir as novas funcionalidades ó sistema que xa temos ata agora. Por isto de agora en adiante tan só se explicarán as novas función ou clases engadidas no prototipo.

A partir da reunión de avaliación do anterior prototipo se obtiveron as novas funcionalidades que debe ter o sistema neste segundo prototipo. Na Figura 25 pódese observar un pequeno diagrama de casos de uso correspondente ás funcionalidades da ferramenta de deseño. E na figura 26 o diagrama de casos de uso da biblioteca para importar a interface nun proxecto externo.

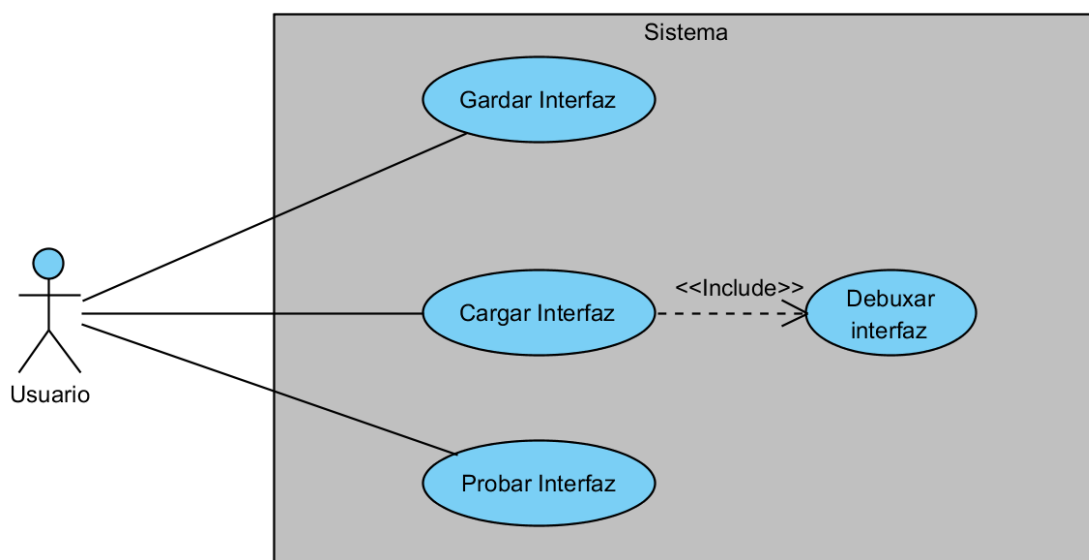


Figura 25. Diagrama de casos de uso da ferramenta de deseño no Segundo prototipo

Gardar interface

Mediante esta función se permitira gardar a interface deseñada na ferramenta nun ficheiro Xml para mais tarde poder utilizalo. Para facer o gardado da interface do ficheiro utilizarase a clase de C# XmlSerializer que permitirá serializar todos os elementos que compoñen a interface e gardalos cos valores das súas propiedades.

Cargar interface

Esta funcionalidade fará posible cargar na interface na nosa ferramenta para continuar co seu deseño. Para cargar o ficheiro coa información da interface tamén se utilizará a clase XmlSerializer que permite deserializar os datos contidos no ficheiro Xml.

Probar Interface

Deberá ser posible facer una previsualización da interface deseñada coma se se estivera a utilizar nun proxecto externo, que é o obxectivo deste sistema. Para isto simplemente se creará unha ventá aparte que tan so conteña a interface tal e como foi deseñada.

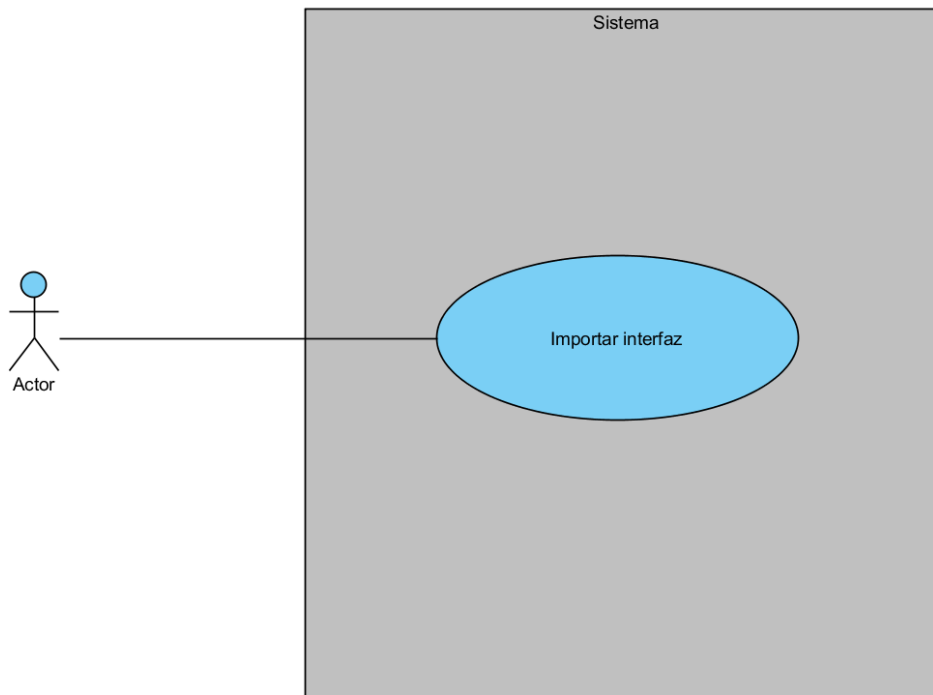


Figura 26. Diagrama de casos de uso da Biblioteca no Segundo prototipo

Importar interface

Esta funcionalidade fai referencia a que a biblioteca debe dar soporte á importación de interfaces dende un ficheiro Xml, nun proxecto C# calquera.

Para entender un pouco mais en detalle o funcionamento dos casos de uso tanto da ferramenta de deseño coma da biblioteca de importación deséñanse os diagramas de secuencia que se poden apreciar a continuación.

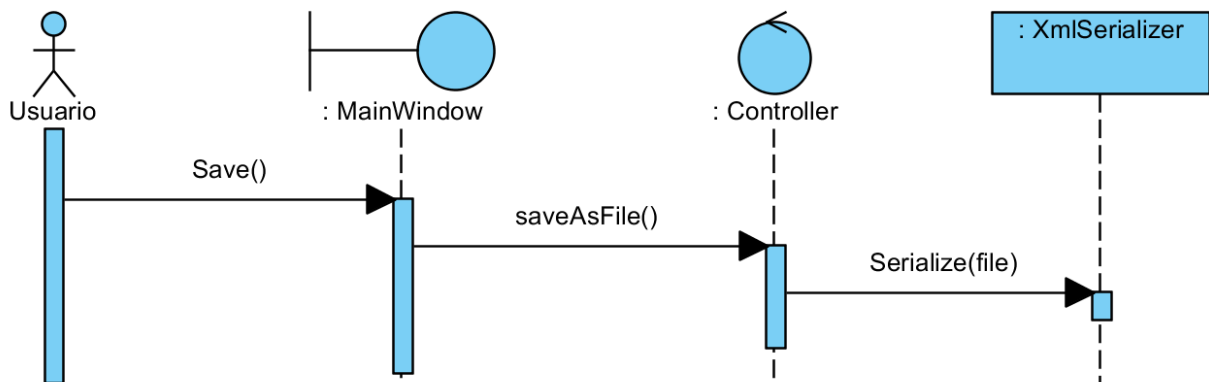


Figura 27. Diagrama de secuencia. Gardar Interface

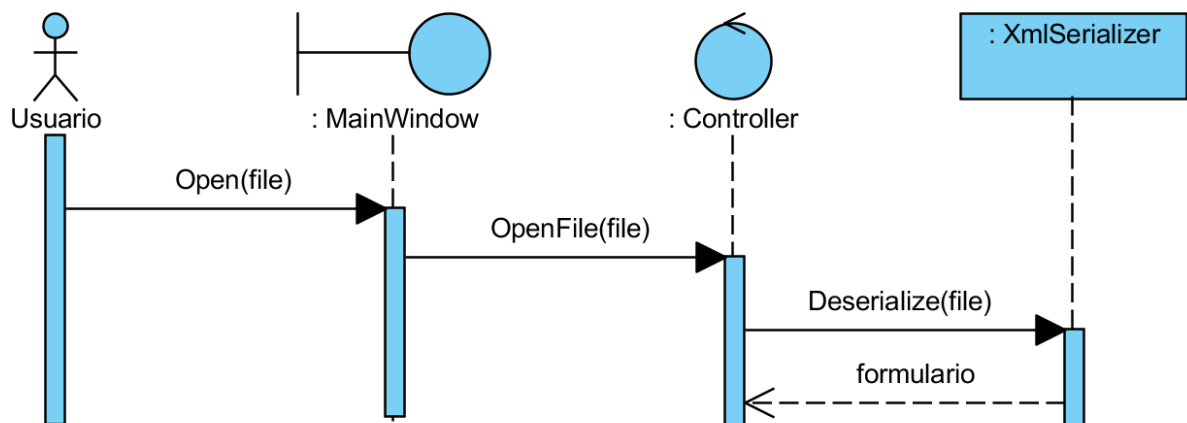


Figura 28. Diagrama de secuencia. Abrir interface

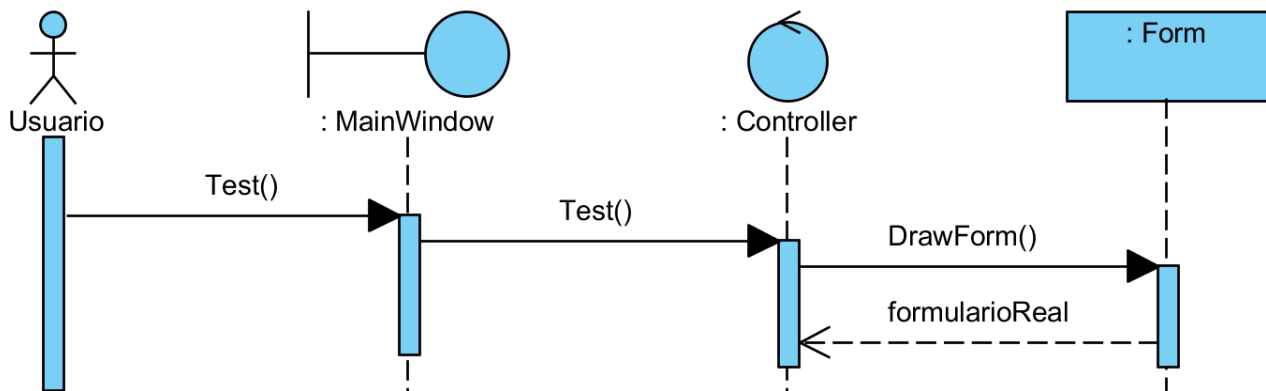


Figura 29. Diagrama de secuencia. Probar interface

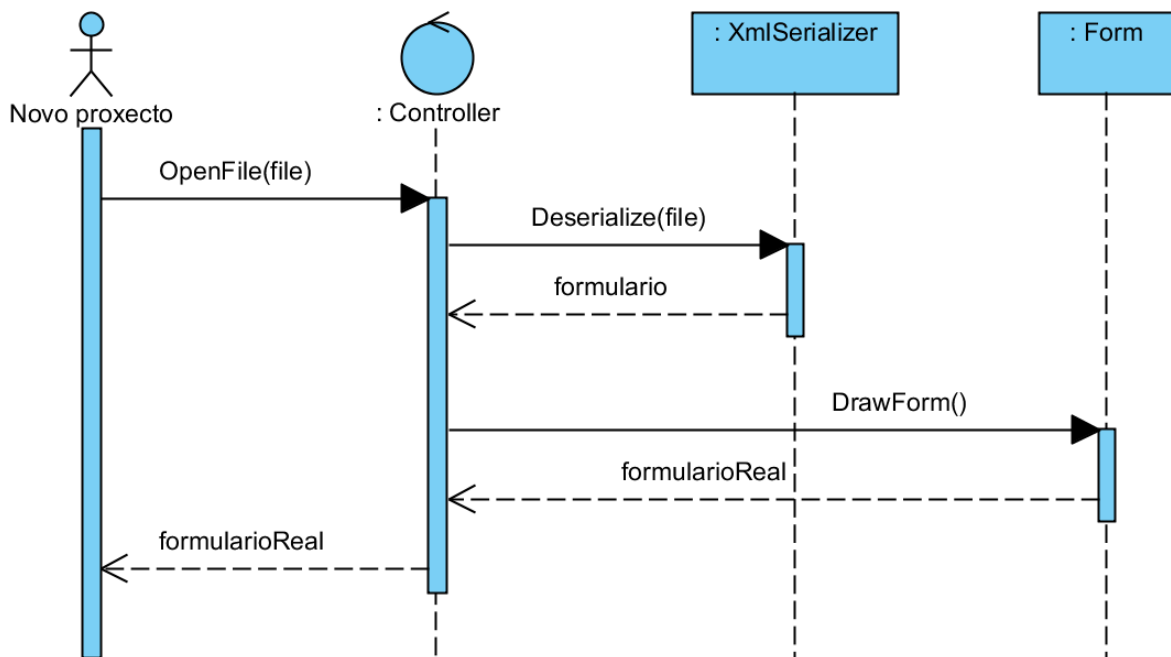


Figura 30. Diagrama de secuencia. Importar interface noutro proxecto

Tras observar estes diagramas e comprender mellor as necesidades do sistema para dar soporte ás funcionalidades que se van a incluír neste prototipo, desenvolveuse un diagrama de clases tanto para a ferramenta de deseño, na Figura 31, como para a biblioteca para facer as importacións noutro proxecto, Figura 32.

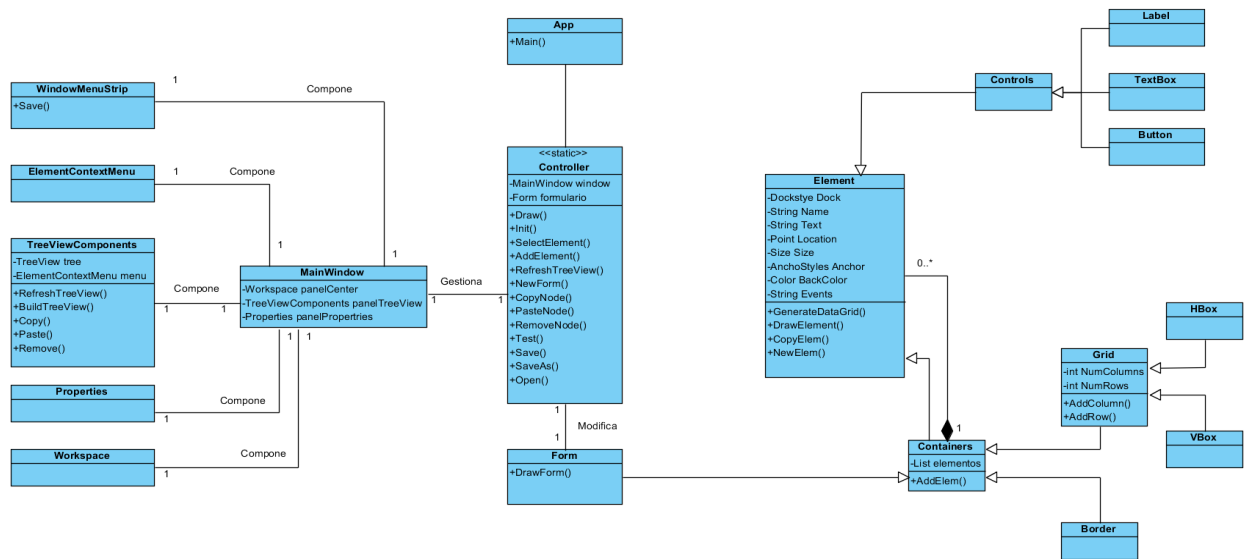


Figura 31. Diagrama de clases do Segundo Prototipo. Ferramenta de deseño

WindowMenuStrip

Esta clase representa o menú superior da ventá e contén botóns para executar as funcións de crear novo formulario, gardar e abrir unha interface, copiar, cortar e pegar elementos e probar e parar proba da interface.

Form

Neste prototipo inclúese unha función fundamental para a funcionalidade de proba da interface, *DrawForm*. Con esta función crease un formulario da API Winforms con todos os controles equivalentes ós elementos que se lle engadiron coa ferramenta. Os eventos son *Strings* que conteñen a acción a realizar cando se dispare o evento.

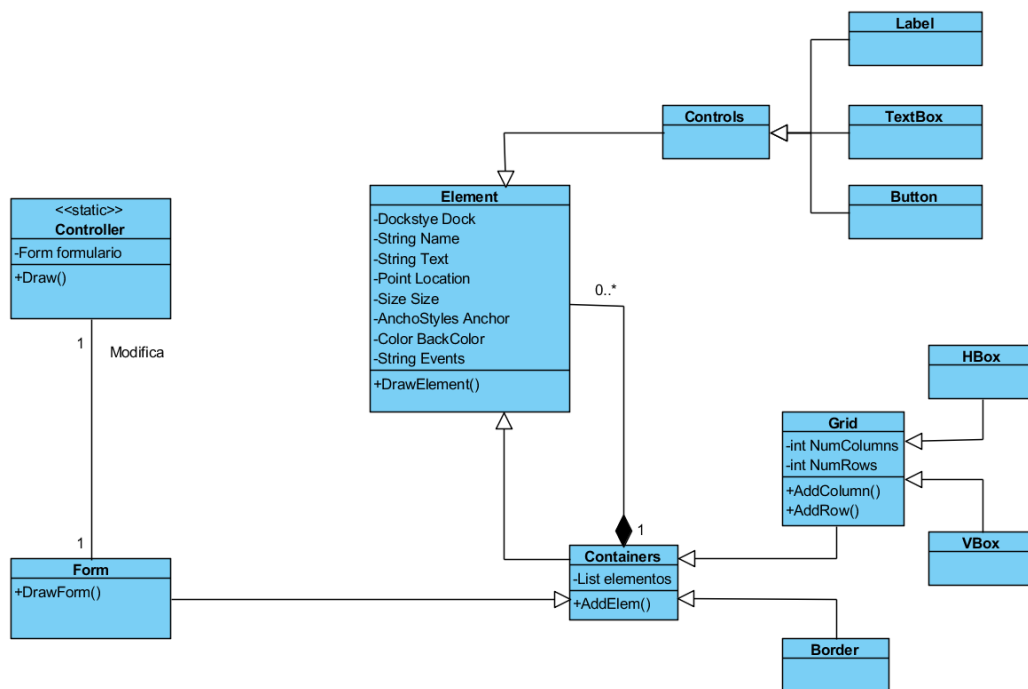


Figura 32. Diagrama de clases do Segundo Prototipo. Biblioteca de importación

Como se pode apreciar no diagrama de clases da biblioteca para importar interfaces nun proxecto, a súa estrutura é idéntica a da ferramenta de deseño. A gran diferenza é a carencia de interface gráfica e as clases unicamente teñen as funcións de debuxado, é dicir, as funcións que xeran os controles da API Winforms.

4.2. Construcción do prototipo

Nesta ocasión, en canto á ferramenta de deseño, a construción do prototipo parte do prototipo anterior. Nel hai que engadirille as novas clases e funcionalidades que se indicaron na fase de deseño. Nesta ocasión engádeselle un menú superior con botóns para executar certas accións no sistema, tales coma copiar, pegar, gardar, etc. Ademais tamén se modifica o panel do marxe da dereita para engadir un novo *DataGridView* que contén os eventos que se permitirán crear coa ferramenta. Tamén engádese a posibilidade de gardar e de abrir interfaces creadas coa ferramenta a través de arquivos .xml. Estes arquivos xeráronse gracias a clase *XmlSerializer* de C#. Esta permite gardar ou ler os valores das propiedades e

atributos de cada un dos elementos que forman da interface. Por último neste prototipo impleméntase a funcionalidade que permitirá probar a interface que se deseñou. Isto faise creando un formulario de Winforms cos controles equivalentes os elementos do noso sistema, e visualizalo creando unha nova ventá aparte da nosa ferramenta.

Con respecto a biblioteca para importar as interfaces nun proxecto externo créase unha nova solución independente da ferramenta de deseño, e créase unha .dll para que poida ser incluída como referencia noutro prototipo e poder importar as interfaces deseñadas. O código desta biblioteca e case totalmente idéntico ó código da ferramenta de deseño eliminando todas as clases que representan á interface gráfica e eliminando todos aqueles métodos ou funcións que non son executados para a creación dun formulario de Winforms.

Como parte final desde prototipo desenvolveuse unhas pequenas probas de funcionamento do sistema de garda e carga de interfaces deseñadas comprobando que tras engadir unha serie de elementos ó formulario ou a outro contenedor, cando se proba a interface esta conteña os elementos que se queira que tiveran.

4.3. Evaluación e retroalimentación do prototipo

Neste segundo prototipo xa se pode apreciar perfectamente como será o sistema final. Na reunión co titor para a segunda demostración do sistema observáronse poucos detalles para mellorar. O mais destacable en canto a melloras é que certos *Containers* deberían inicializarse cun valor para a propiedade *Dock* concreta, o *Border* debe iniciarse con esa propiedade a *Fill* o mesmo que o *Grid* e o *VBox*, e o *HBox* debe iniciarse co valor *Top*.

Decidiuse que no seguinte prototipo debería permitirse a creación de eventos nos elementos que se engadan na interface, polo que a biblioteca tamén debe soportar esta nova funcionalidade.

Ademais é o momento de ampliar o catálogo de elementos a engadir de forma que xa se inclúan todos os elementos que se especificaron nos obxectivos do proxecto.

En cuestión de novas funcionalidades neste prototipo xa se cubriron todas as que se planificaran neste proxecto polo que o seguinte prototipo simplemente é para ampliar a

variedade de controles e engadir algunha propiedade os elementos xa existentes.

5. Tercer prototipo

5.1. Deseño rápido

Este prototipo ten como obxectivo principal completar os obxectivos do proxecto que faltaban, estes son, completar o catálogo de elementos e permitir o control de eventos dos controles. Cubrir estes novos engadidos fai que a estrutura da aplicación creza considerablemente xa que canda control novo supón unha nova clase. Ademais hai que ter en conta que existen unha serie de elementos un tanto especiais, algúns deles, a pesar de non ser elementos contedores, posúen subelementos. Estas ampliacións da estrutura do sistema pódense apreciar na Figura 33.

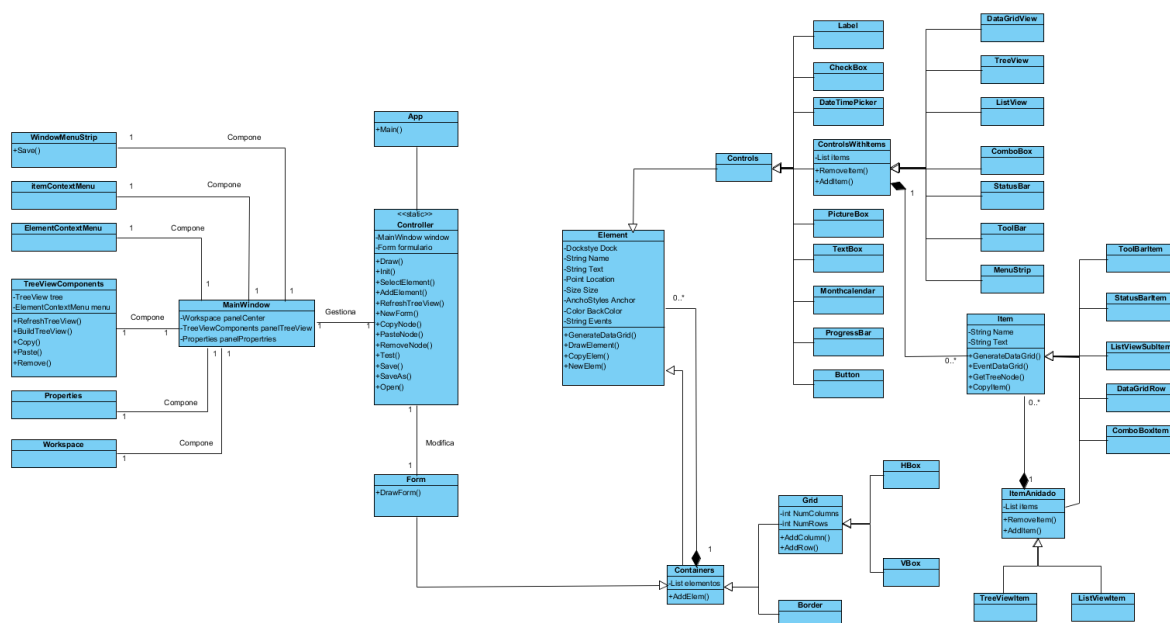


Figura 33. Diagrama de clases do Terceiro prototipo. Ferramenta de deseño

Element

Nesta clase inclúese as propiedades correspondentes ós eventos. No diagrama se indican

como *String events*, aínda que en realidade son unha serie de *Strings* sendo un por cada evento soportado.

- **Atributos**
 - Conxunto de *Strings* que almacenan as instrucións que se executarán ó lanzarse o evento correspondente
- **Métodos**
 - *EventDataGrid*: Esta función xera a táboa na que se poderá indicar as instrucións a realizar no evento indicado.

ControlsWithItems

Esta clase representa os elementos que posúen subelementos pero sen ser elementos contedores. Por exemplo é o caso do *TreeView* que posee nodos. Desta clase estenden os elementos **DataGridView**, **TreeView**, **ListView**, **ComboBox**, **StatusBar**, **ToolBar** e **MenuStrip**.

- **Atributos**
 - *List<Item> items*: Lista que contén os ítems que forman parte do elemento.
- **Métodos:**
 - *AddItem*: Función que engade un subelemento ó elemento en cuestión.
 - *RemoveItem*: Función que elimina un subelemento do elemento seleccionado.

Item

Esta clase fai referencia ós subelementos que forman parte de controles que estenden da clase *ControlsWithItems*. Desta clase estenden as clases **ToolBarItem**, **StatusBarItem**, **ListViewSubItem**, **DataGridRow** e **ComboBoxItem**.

- **Atributos**
 - Os atributos desta clase correspóndense coas propiedades dos Controles da API Winforms equivalentes ós Items. Neste caso tan só se permite editar o Name e o Text.
- **Métodos**
 - *GenerateDataGrid*: Con esta función xérase a táboa onde se poden editar as propiedades dos *Items*.

- *EventDatagrid*: Esta función xera a táboa onde se poden editar os eventos dos *Items*.
- *GetTreeNode*: Esta función servirá para xerar a sección de *TreeView* correspondente ó *Item* no panel esquerdo da interface da ferramenta.
- *CopyItem*: Con este método se poderá copiar un *Item* que forma parte dun *Element* de forma que cando este se copie se copie con todos os *Items* que o forman.

ItemAnidado

Esta clase representa a aqueles *Items* que á súa vez conteñen ítems tamén, como poden ser os nodos do *TreeView* que tamén poden ter subnodos. Desta clase estenden os *Items TreeViewItem* e *ListViewItem*.

- **Atributos**
 - *List<Item> items*: Lista que contén os *Items* que forman parte do elemento.
- **Métodos:**
 - *AddItem*: Función que engade un subelemento ó elemento en cuestión.

En canto á estrutura da biblioteca para a importación de interfaces, xeradas coa ferramenta de deseño, noutro proxecto en desenvolvemento, pódese ver no diagrama de clases da Figura 34 como é idéntica á estrutura da ferramenta de deseño, tendo coma diferenza a inexistencia de interface gráfica e de métodos que non sexan para xerar os controles da API Winforms equivalentes ós elementos.

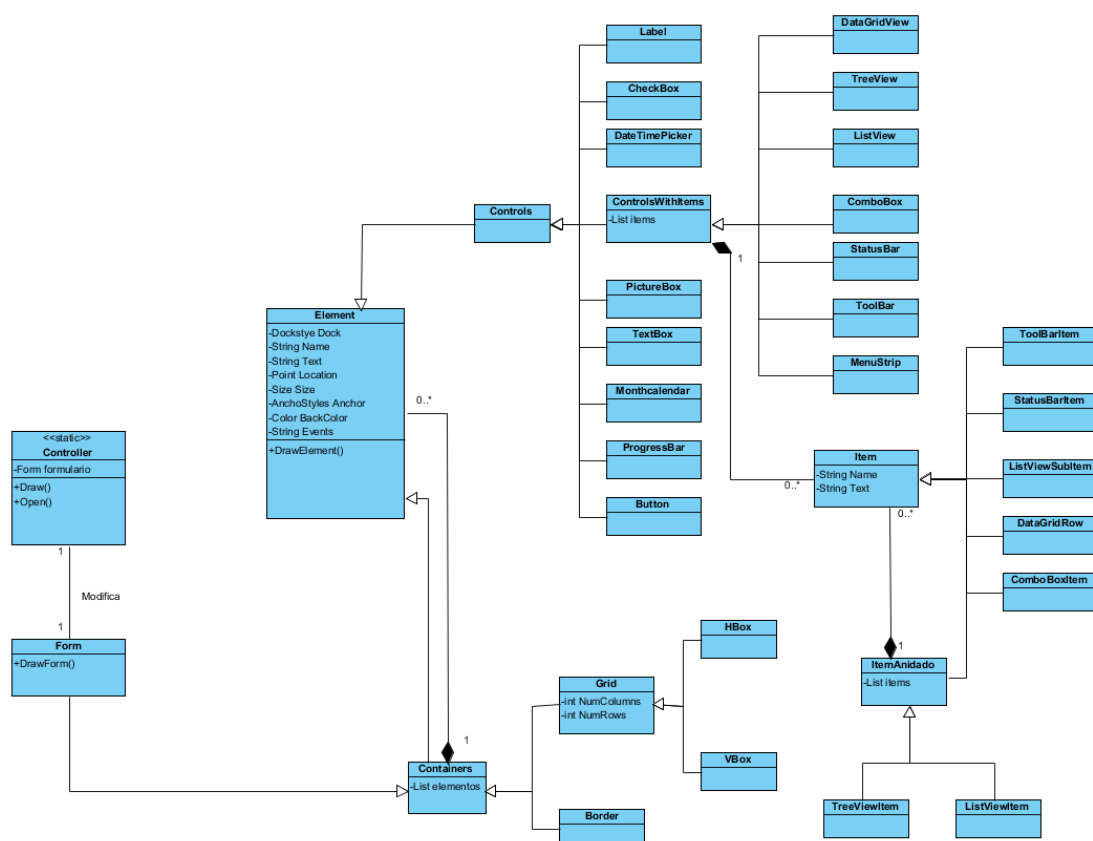


Figura 34. Diagrama de classes do Terceiro prototipo. Biblioteca par importar interfaces

5.2. Construcción do prototipo

Finalmente imos construír o último prototipo planificado neste proxecto. Nesta ocasión aumentou o número de elementos que se lle poden engadir a unha interface deseñada con esta ferramenta, polo que a implementación deste prototipo consistiu maiormente en crear novas clases.

Engádesse un novo subtipo de *Control* que é o *ControlItems* e se refire a aqueles elementos que non son contedores pero si teñen subelementos no seu interior. Codificouse coma unha clase abstracta que herda de *Control* e que permite tratar todos os elementos deste tipo de forma totalmente transparente.

Engádese a clase *Item*. Dela estenden os elementos que forman parte dos elementos que estenden de *ControlItems* coma *TreeView* ou *ComboBox*. A clase *Item* implementouse coma

unha clase abstracta moi similar á clase *Element*. A clase *Item* posúe os métodos virtuais *EventDataGrid* e *GenerateDataGrid* que reduce a cantidade de código de novos elementos que podan entender esta clase facendo moito mais sinxelo engadilos. Tamén implementa a función *GetTreeNode* que é común a todas as posibles clases que poidan herdar dela. Por ultimo está presente a función abstracta *CopyItem* que a deben codificar cada unha das clases que herden de *Item*.

Ademais herdando de *Item* créase unha nova clase abstracta que é *ItemAnidado* para definir aqueles *Items* que a súa vez posúen subitems. Unicamente diferénciase da clase *Item* en que posúe unha lista de subitems, polo demais posúe os mesmos métodos abstractos que as clases que herden del deben implementar.

Finalmente desenvolvéronse probas de unidade para todas as novas clases que herdan de *Control*, *ControlItems*, *Item*, *ItemAnidado* e *Container*. Estas probas comprobaban o correcto funcionamento da implementación de todos os métodos abstractos de forma que se se fan modificacións nun futuro, estas probas permitirán comprobar que as modificacións non afectaron o correcto funcionamento do método.

5.3. Evaluación e retroalimentación do prototipo

Finalmente faise a reunión de avaliación do terceiro prototipo. Nela compróbase o funcionamento da ferramenta de deseño para estar seguro de que cubre todos os obxectivos e funcionalidades que se esperan do proxecto. Efectivamente así é, o mesmo que a biblioteca para a importación de interfaces en un proxecto C# axeo a este sistema. Polo tanto este último prototipo queda listo para convertelo no produto final.

6. Desenvolvemento do produto

Todas as demostracións de funcionamento do sistema fixéronse baixo un sistema operativo GNU/Linux e en ningunha ocasión ata agora foi nun sistema Windows. Tras probalo neste

sistema démonos de conta que non funcionaba da forma que se esperaba polo que era necesario corrixir certos detalles da implementación que facían que o sistema desenvolvido fora incompatible con Windows. A pesar deste problema non se considerou preciso realizar un novo prototipo xa que se considerou simplemente coma unha pequena fase de refinamento xa que os problemas encontrados non requiriron unha extensa implementación, se non que unicamente a modificación pequenos detalles. Entre eles podemos destacar que mentres que en GNU/Linux permite encher un *ComboBox* dunha celda do *DataGridView* de propiedades con *enums*, en windows daba error o seleccionar un deles, polo que houbo que substituír os valores dos *enums* por *Strings* co mesmo valor. Outro problema encontrado foi ó crear a interface gráfica da ferramenta. O crearse lanzábase unha excepción debido a que se creaban obxectos do formulario por separado e logo se lanzaba a aplicación. Foi curioso e inesperado toparse coa existencia destas diferencias, pero tamén valeu de cara o futuro para ter en conta este tipo de imprevistos.

Tras corrixir os pequenos problemas encontrados na demostración do terceiro prototipo, para facelo compatible ó 100% con Windows e GNU/Linux simultaneamente, e como xa se cubriron todos os obxectivos e funcionalidades planificadas para o proxecto, é o momento de converter o último prototipo no produto final.

MANUAL DE USUARIO

A aplicación para o deseño de interfaces gráficas da API Winforms de .Net y Mono, é unha aplicación que proporciona unha alternativa gráfica á programación de interfaces de usuario no linguaxe C# e a API mencionada.

Trátase dunha ferramenta pensada para programadores C# tanto en sistemas GNU/Linux coma Windows. Esta aplicación permite que de forma sinxela, rápida e gráfica desenvolver a parte visual dun proxecto C# facendo ó mesmo tempo que se separe a capa de presentación da lóxica da aplicación.

Nos seguintes capítulos que forman parte deste manual especificarase toda a información necesaria para o uso desta ferramenta incluíndo os requisitos previos e os posibles erros que se podan atopar.

1. Requisitos Hardware e Software

En canto ós *requisitos hardware*, esta ferramenta non presenta ningunha restrición especial, puidéndose executar en calquera sistema que posúa o software necesario, que a continuación especificaremos.

En relación ós *requisitos software*, precísanse da existencia dunha serie de produtos software:

- A aplicación pódese obter a través do CD adxunto a esta documentación ou a través dun repositorio de software remoto (<https://github.com/aalvlopez/generadorWinforms>). Para utilizar a opción do depositario remoto débese ter instalado git. Para a instalación do mesmo nun sistema GNU/Linux baseado en Debian débese executar o seguinte comando:

```
sudo apt-get install git
```

Se se quere facer a clonación do repositorio nun sistema windows ou noutra distribución GNU/Linux débese igualmente instalar git descargándoo da súa páxina oficial <http://git->

scm.com/download

- Unha vez temos á aplicación no noso sistema precísase ter instalado o framework correspondente. No caso dun sistema Windows precísase a versión 4 do framework .Net dispoñible na páxina oficial de microsoft (www.microsoft.com).

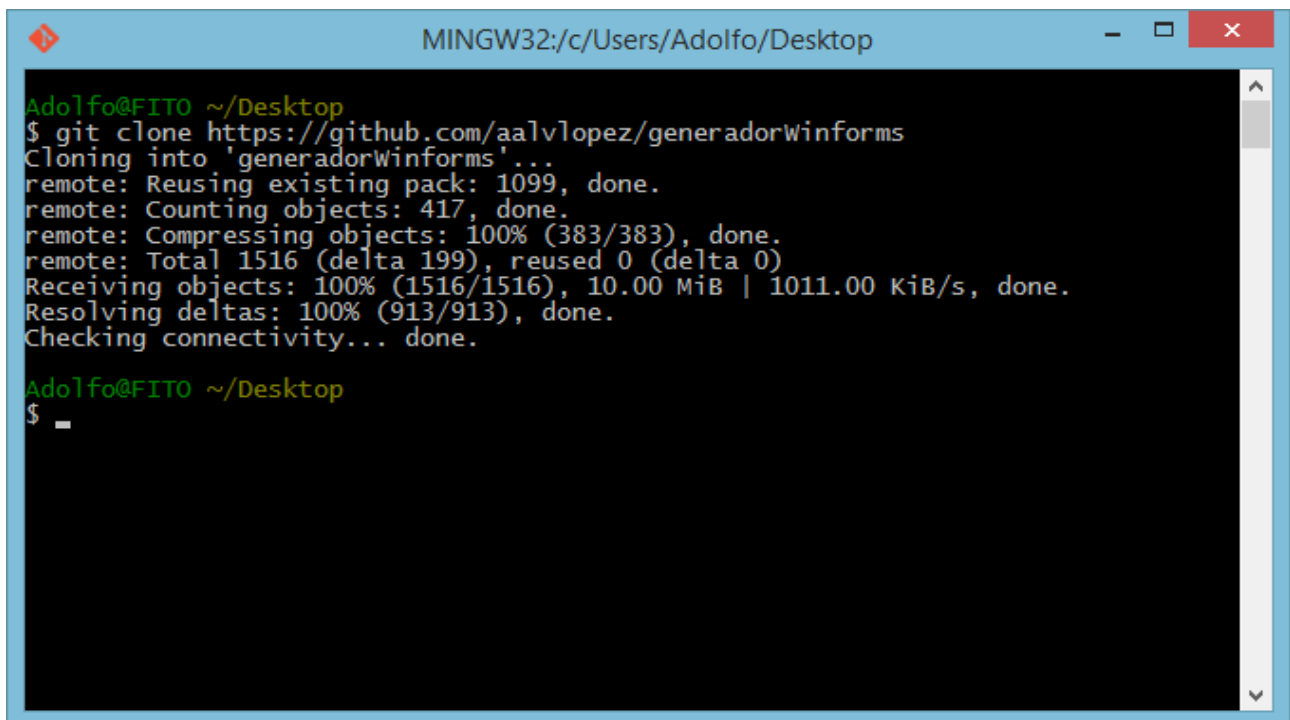
Se polo contrario estase a utilizar unha distribución GNU/Linux débese instalar o framework Mono dispoñible para descargar dende a súa páxina oficial (<http://www.mono-project.com/>) ou no caso dunha distribución baseada en Debian mediante o comando:

```
sudo apt-get install mono-complete
```

2. Execución

Windows

Primeiramente debemos clonar o proxecto do repositorio ou introducir o CD adxunto á esta documentación. Para a clonación utilizaremos a aplicación *git* que instalamos previamente e clonaremos do repositorio <https://github.com/aalvlopez/generadorWinforms>.

A screenshot of a Windows command prompt window titled "MINGW32:/c/Users/Adolfo/Desktop". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The command prompt shows the following text:

```
Adolfo@FITO ~/Desktop
$ git clone https://github.com/aalvlopez/generadorWinforms
Cloning into 'generadorWinforms'...
remote: Reusing existing pack: 1099, done.
remote: Counting objects: 417, done.
remote: Compressing objects: 100% (383/383), done.
remote: Total 1516 (delta 199), reused 0 (delta 0)
Receiving objects: 100% (1516/1516), 10.00 MiB | 1011.00 KiB/s, done.
Resolving deltas: 100% (913/913), done.
Checking connectivity... done.

Adolfo@FITO ~/Desktop
$ _
```

Figura 35. Clonación do repositorio en windows

Unha vez feita a clonación do repositorio dirixímonos ó directorio creado, “generadorWinforms” e accedemos ó directorio “Aplicación final”. Nela poderemos atopar dúas carpetas, unha delas porá “Windows”. Entrando nela atoparemos o instalador da aplicación.

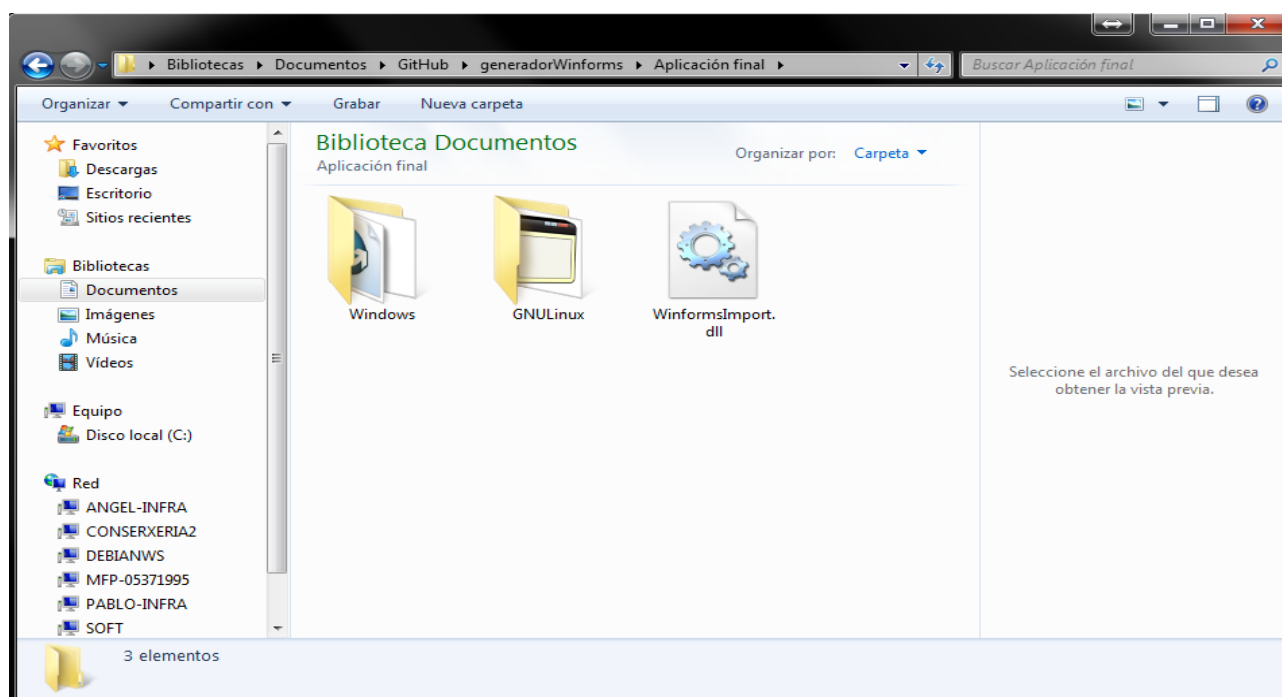


Figura 36. Contenido del directorio "Aplicación final"

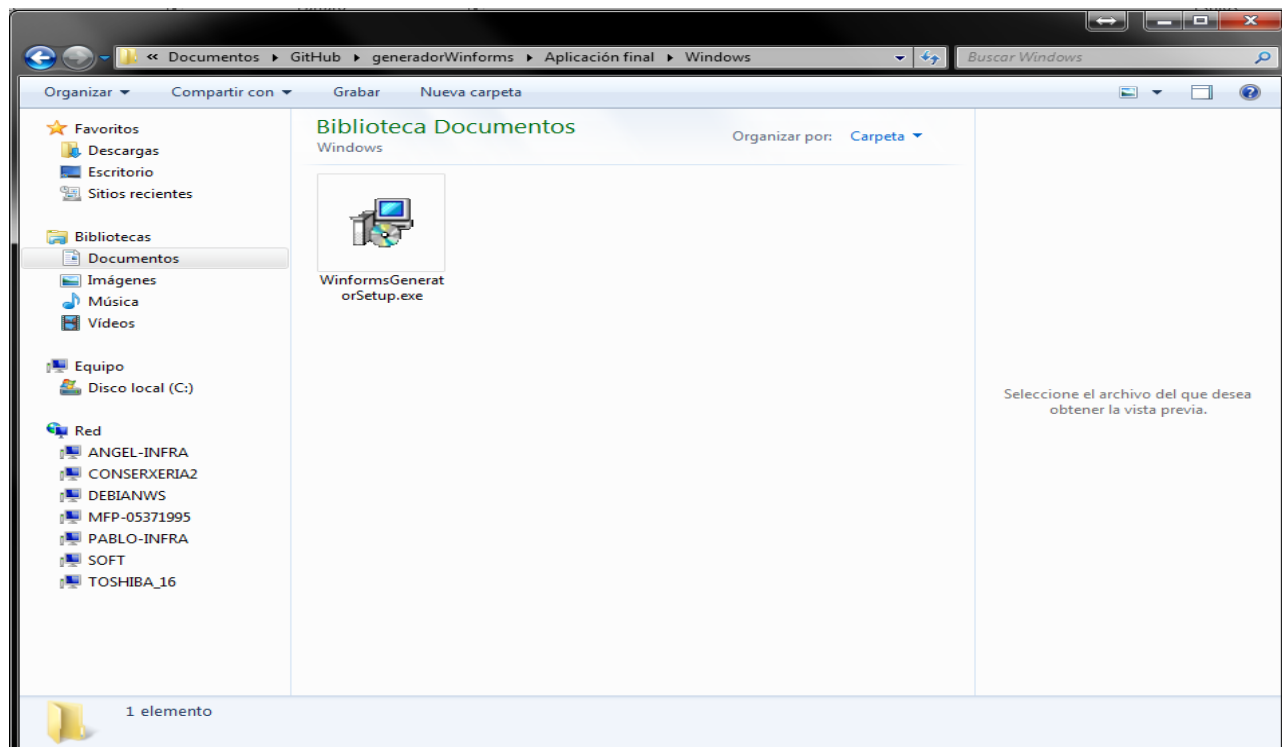


Figura 37. Instalador en Windows

Para executar tan só é preciso facer dobre click sobre o executable e este xa lanza a aplicación de instalación.

Primeiramente deberase seleccionar o idioma de instalación como se indica na Figura 38.

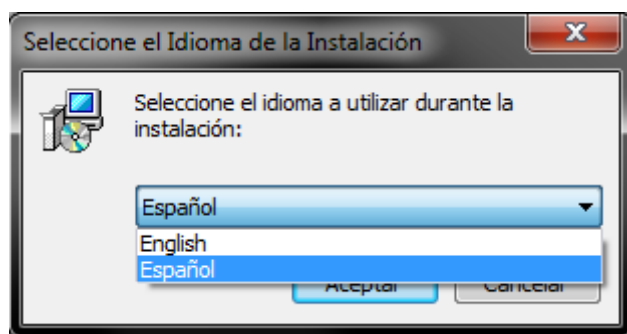


Figura 38. Selección do idioma

Tras isto continuarase navegando a través das distintas pantallas onde se dará a posibilidade de elixir a carpeta de instalación, a creación ou non dunha carpeta no menú de inicio e a elección de se crear un icono no escritorio.

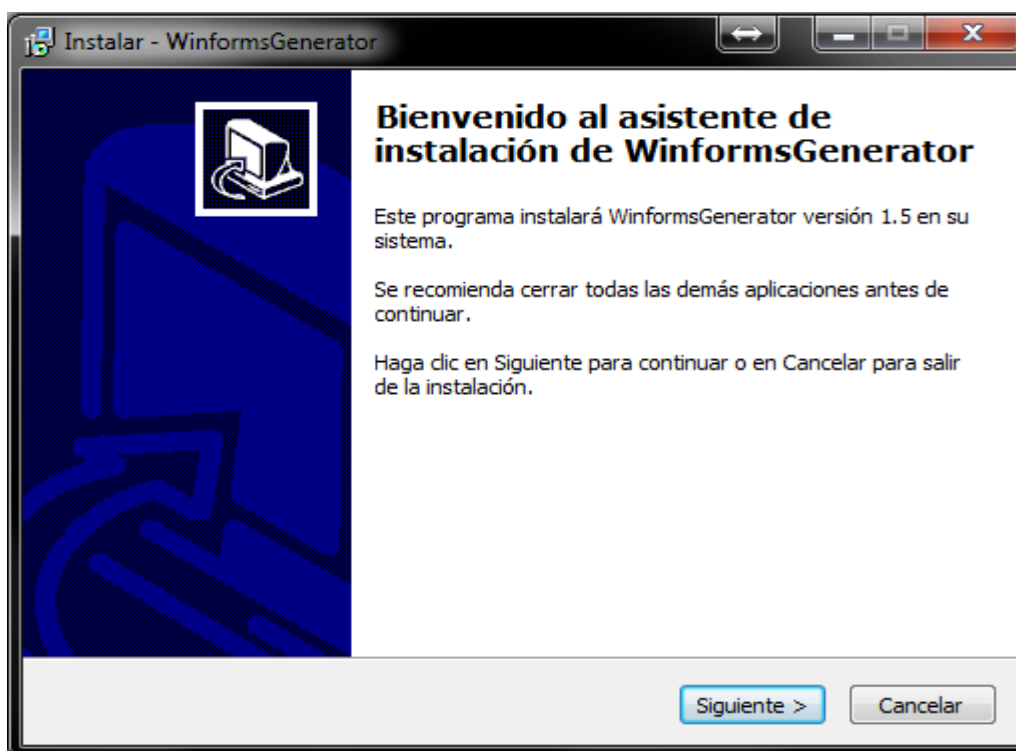


Figura 39. Comezando a instalación da aplicación

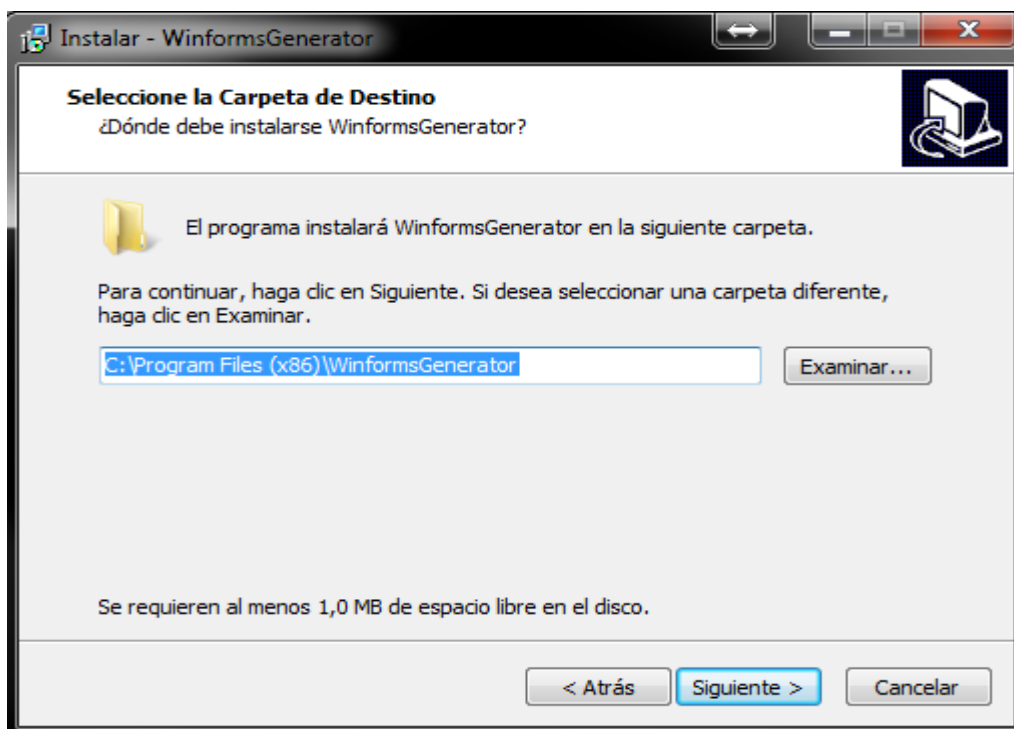


Figura 40. Elección da carpeta de instalación

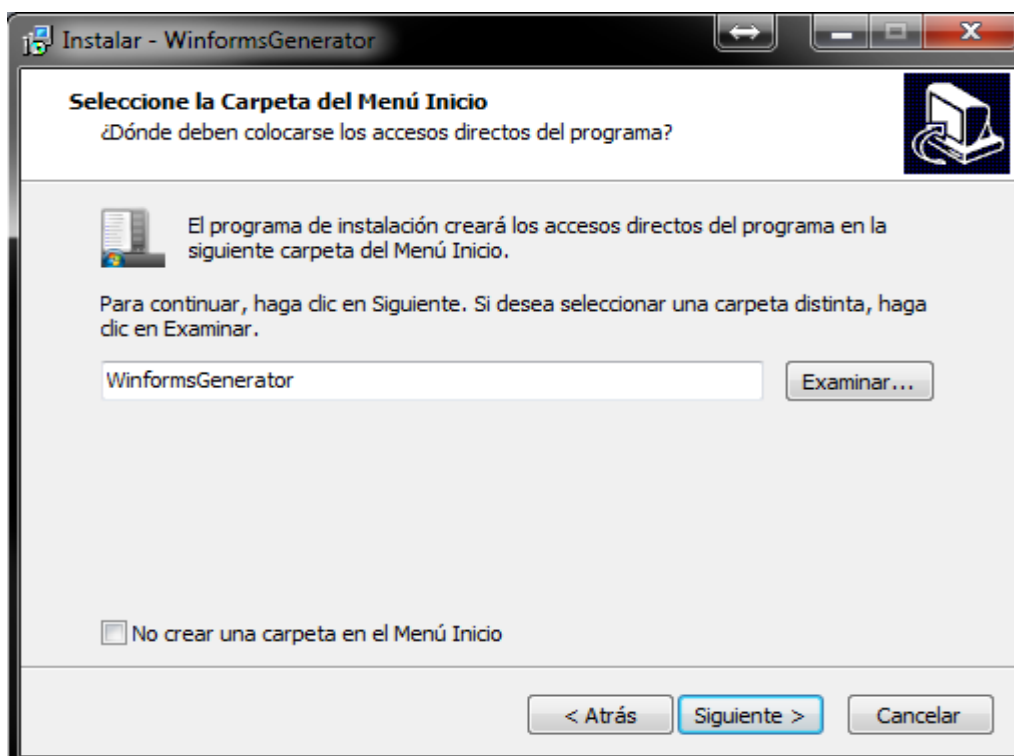


Figura 41. Menú de inicio

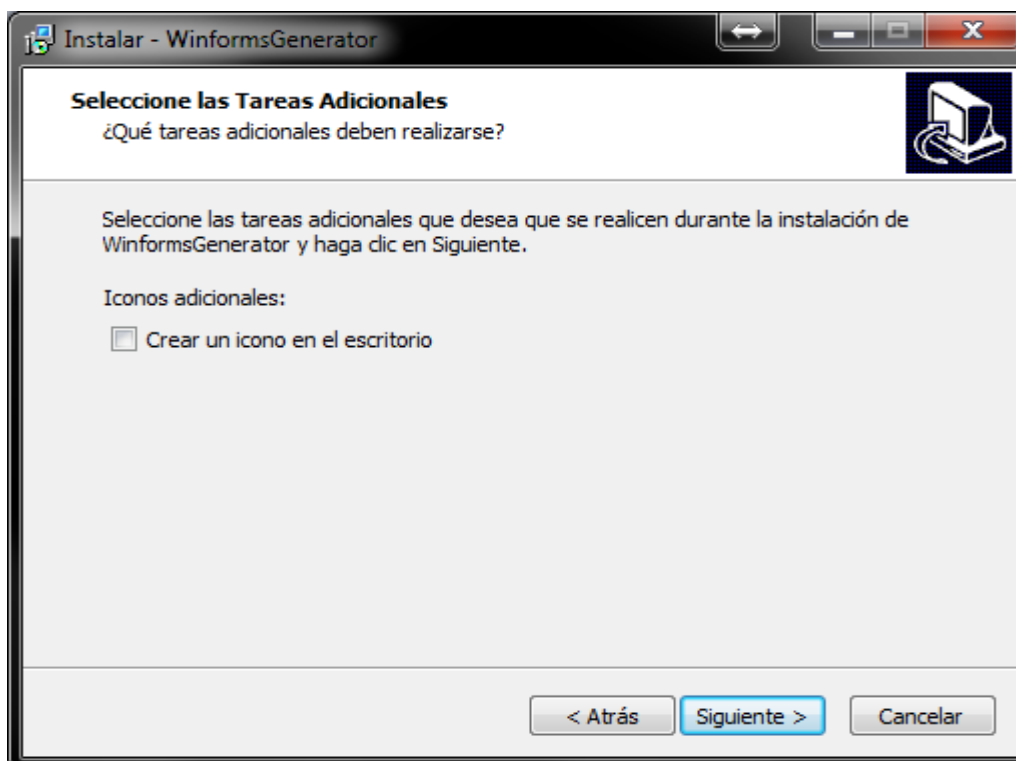


Figura 42. Icono no escritorio

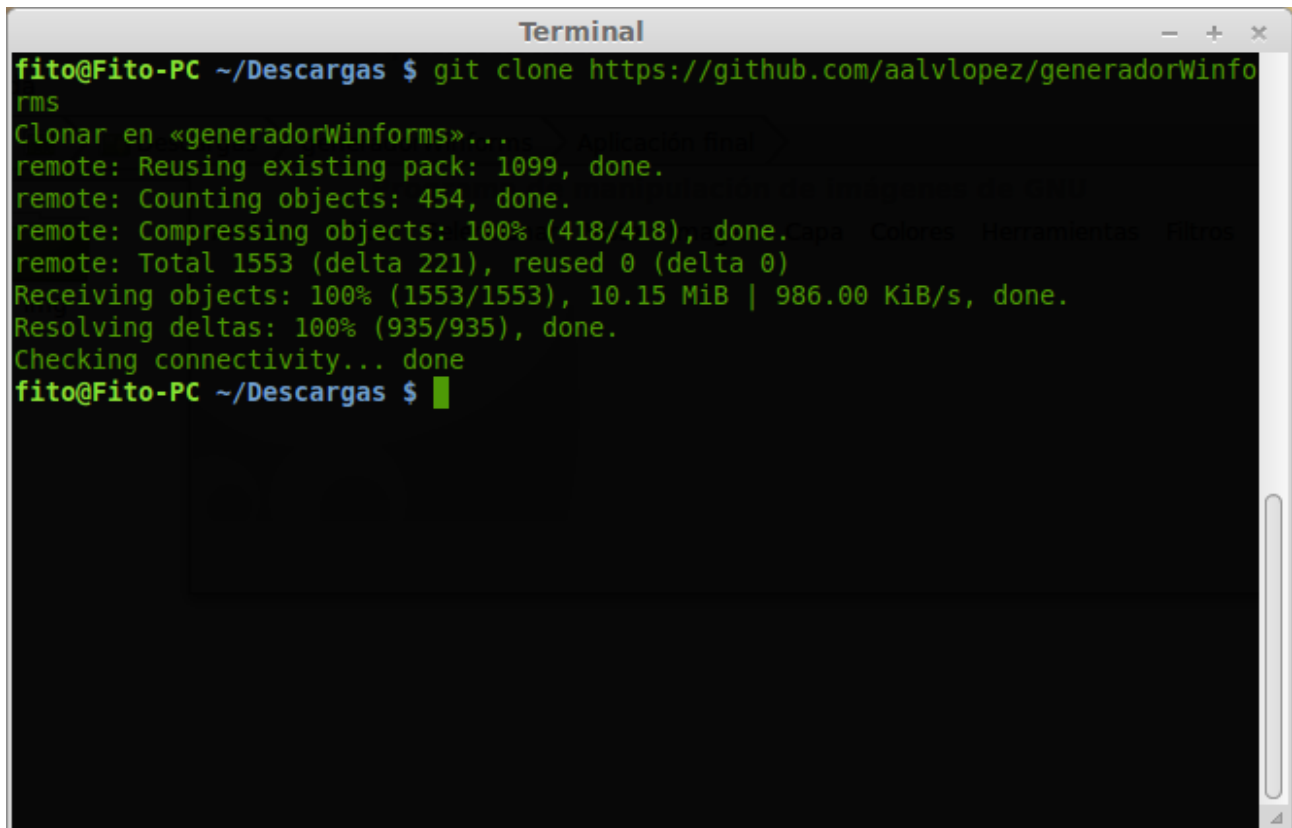


Figura 43. Fin e resume da instalación.

Unha vez finalizada a instalación a aplicación estará dispoñible para ser utilizada.

GNU/Linux

Neste caso utilizamos unha distribución baseada en Debian. Tendo xa instalado o software preciso facemos a clonación da mesma forma ca en Windows.

A screenshot of a Linux terminal window titled "Terminal". The prompt is "fito@Fito-PC ~/Descargas \$". The user has entered the command "git clone https://github.com/aalvlopez/generadorWinforms". The terminal output shows the progress of cloning the repository: "Clonar en «generadorWinforms»...", "remote: Reusing existing pack: 1099, done.", "remote: Counting objects: 454, done.", "remote: Compressing objects: 100% (418/418), done.", "remote: Total 1553 (delta 221), reused 0 (delta 0)", "Receiving objects: 100% (1553/1553), 10.15 MiB | 986.00 KiB/s, done.", "Resolving deltas: 100% (935/935), done.", and "Checking connectivity... done". The prompt is now "fito@Fito-PC ~/Descargas \$" with a cursor. There are some faint, illegible text artifacts in the background of the terminal output.

```
fito@Fito-PC ~/Descargas $ git clone https://github.com/aalvlopez/generadorWinforms
Clonar en «generadorWinforms»...
remote: Reusing existing pack: 1099, done.
remote: Counting objects: 454, done.
remote: Compressing objects: 100% (418/418), done.
remote: Total 1553 (delta 221), reused 0 (delta 0)
Receiving objects: 100% (1553/1553), 10.15 MiB | 986.00 KiB/s, done.
Resolving deltas: 100% (935/935), done.
Checking connectivity... done
fito@Fito-PC ~/Descargas $
```

Figura 44. Clonación en Linux

Unha vez feita accedemos ó directorio creado en nel ó directorio “Aplicación final” onde encontraremos como xa explicamos o directorio “img” e o executable.

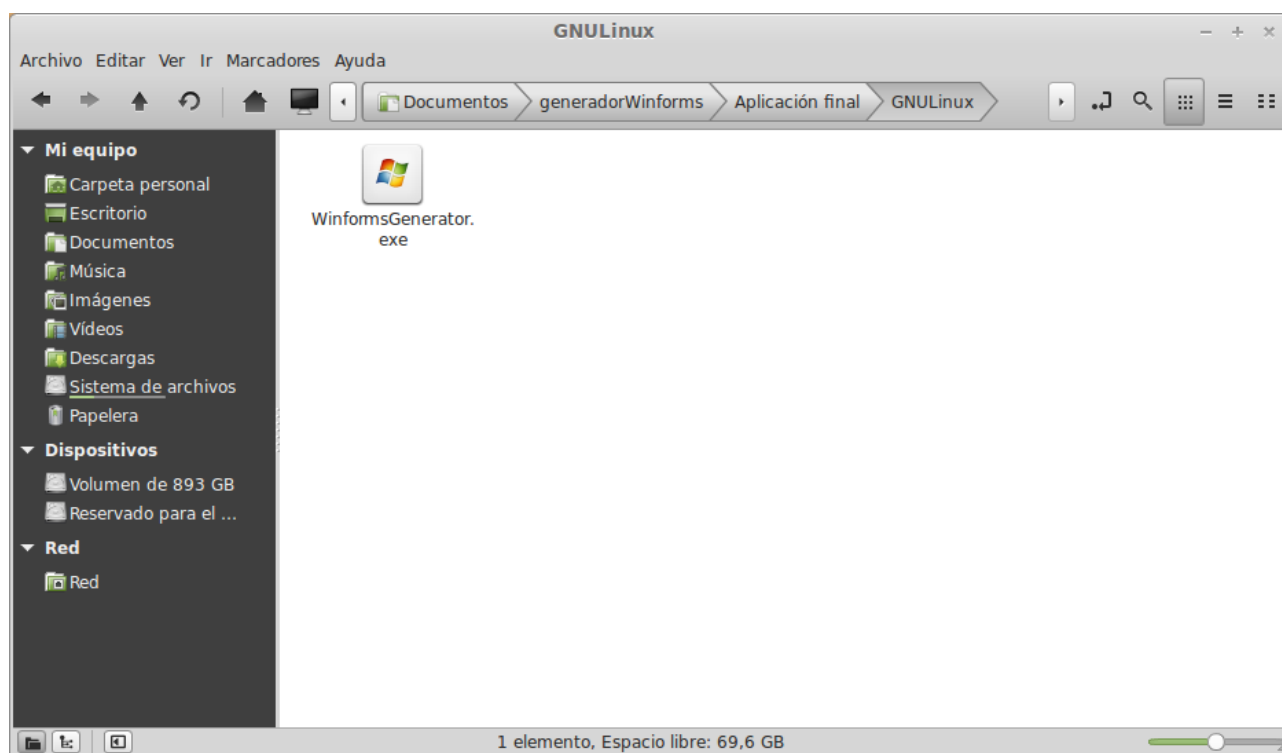


Figura 45. Contenido del directorio en Linux

Para lanzar la aplicación hacemos click derecho para abrir el menú contextual e elegimos la opción “abrir con” e dentro elegimos “Mono Runtime”. No caso de que esa sea la aplicación que abra este tipo de archivos de forma predeterminada únicamente sería preciso hacer doble click.

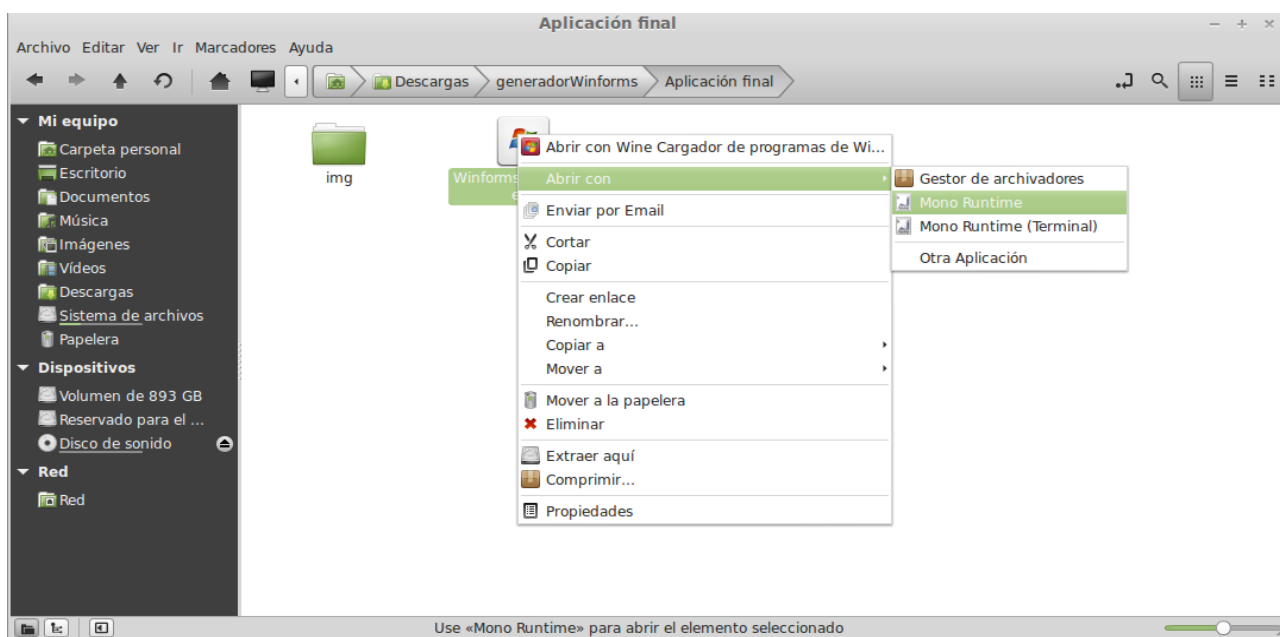


Figura 46. Ejecutar a aplicación en linux.

3. Funcionamiento

Unha vez executada a aplicación debería aparecer unha interface similar á que se pode apreciar na Figura 47.

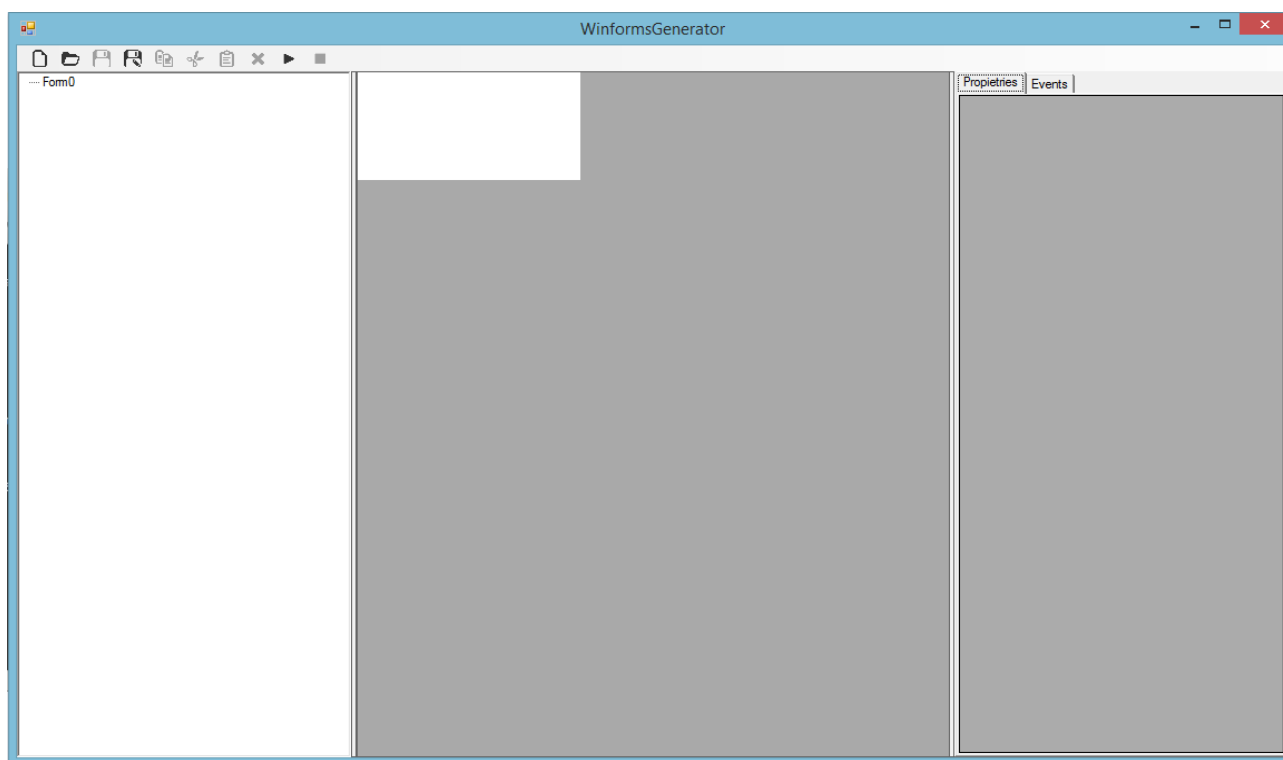


Figura 47. Aplicación en funcionamiento.

Esto correspóndese coa aplicación de diseño de interfaces. Nesta sección falaremos de como utilizar esta ferramenta, que elementos podemos utilizar para crear a nosa interface e como configuralos, e como exportar a nosa interface. Tamén falaremos de como utilizar estas interfaces creadas nun proxecto C#.

3.1. A ferramenta de deseño.

3.1.1. A interface

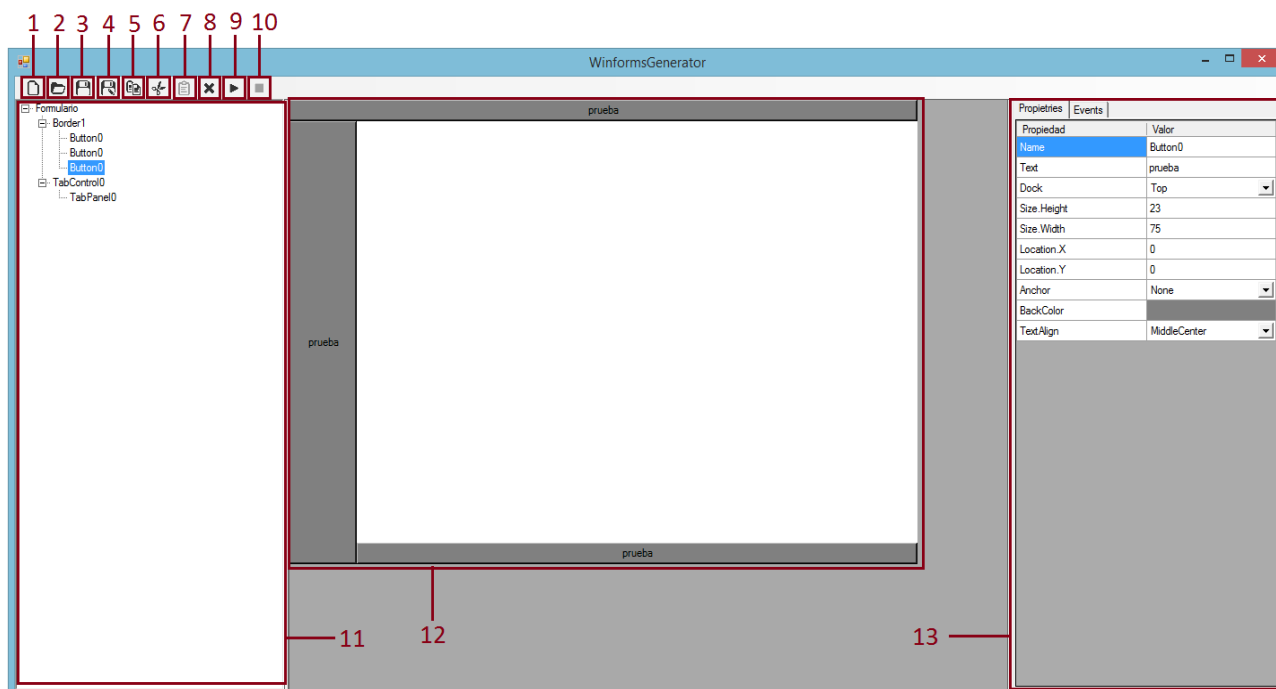


Figura 48. A interface a aplicación.

- **Novo formulario:** Facendo click neste botón xérase un novo formulario perdendo os avances non gardados da interface que se estivera a deseñar.
- **Abrir:** Con esta opción ábrese un explorador de arquivos para que elixas un arquivo Xml que conteña a especificación dunha interface xerada coa mesma ferramenta.
- **Gardar:** Con esta opción gárdanse os progresos da aplicación. Se nos se executou aínda a función de Gardar como esta opción esta debilitada.
- **Gardar como:** Permite gardar a interface que se está a deseñar na ruta que desexes a través dun explorador de arquivos.
- **Copiar:** Copia o elemento que estea seleccionado na árbore de elementos e habilitase a opción pegar.
- **Cortar:** Corta o elemento que estea seleccionado na árbore de elementos e

habilitase a opción pegar.

- **Pegar:** Pega o elemento copiado previamente como fillo do elemento seleccionado na árbore de elementos.
- **Eliminar:** Elimina de forma permanente o elemento que este seleccionado no árbore de elementos.
- **Probar:** Con esta opción ábrese nunha nova ventá a interface creada para que poidas ver dunha forma mais real a interface deseñada.
- **Parar proba:** Con esta función péchase a ventá de proba.
- **Árbore de elementos:** Nesta sección da interface pódese ver un árbore que representa todos os elementos engadidos á interface e a súa posición.
- **Zona de traballo:** Nesta área da aplicación poderase observarse a construción da interface que se estea a deseñar.
- **Táboa de propiedades:** esta zona está formada por dúas pestanas, unha de propiedades e outra de eventos. A táboa de propiedades contén as propiedades que se poden editar dun elemento, e a táboa de eventos contén os eventos que se poden configurar do elemento seleccionado.

3.1.2. O formulario

Cando se abre a aplicación ou se executa a función de *New Form* a aplicación crea un novo formulario en branco. O formulario é o contedor principal de toda a interface, é o nodo principal do árbore de elementos.

As propiedades modificables do formulario son:

- **Name:** Trátase do nome identificador do obxecto.
- **Text:** Correspóndese co título da ventá que se xerará.
- **Size.Height:** Define a altura do formulario.
- **Size.Width:** Define o ancho do formulario.
- **BackColor:** Permite cambiar o color de fondo do formulario.

Como xa se dixo este é o contedor principal de toda a interface que se vai deseñar, polo que

non é posible nin borrarlo nin copialo nin cortalo.

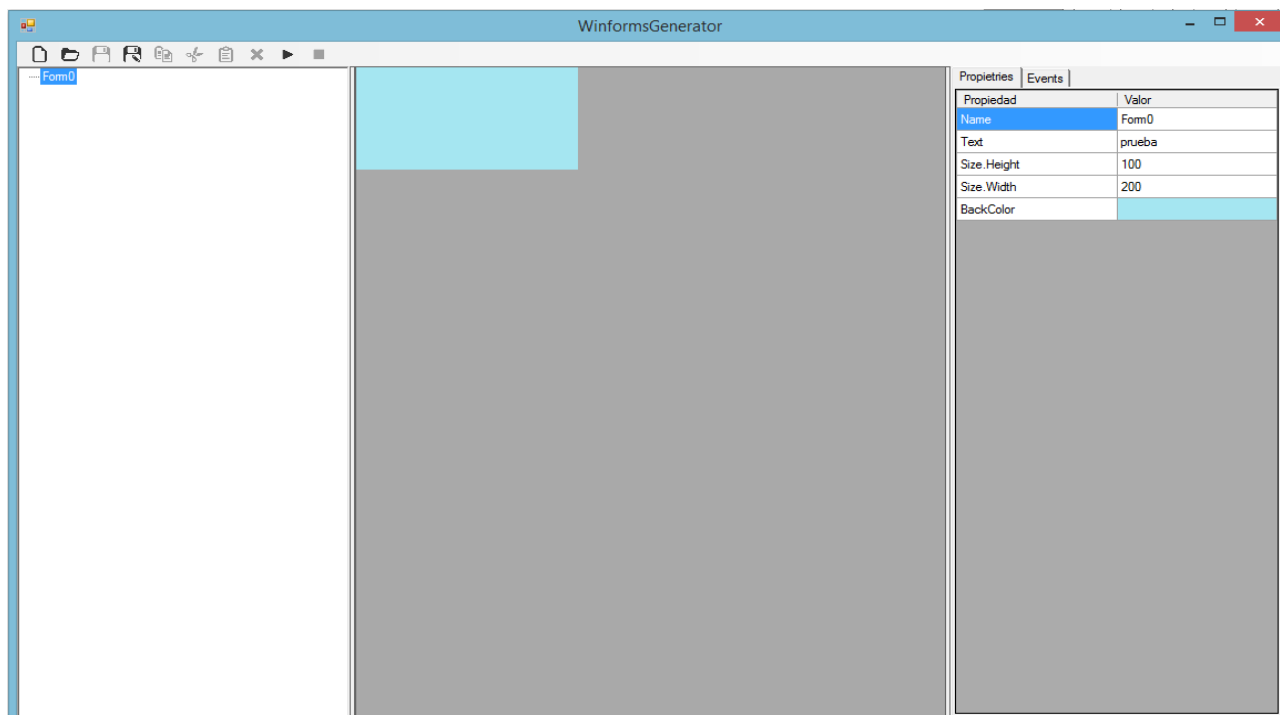


Figura 49. O formulario.

3.1.3. Os elementos

Para deseñar a nosa interface están dispoñibles un conxunto de elementos, tanto do tipo contedor coma elementos simples. Cada un destes elementos posúe unha serie de propiedades que están dispoñibles para editar e así personalizar o aspecto dos elementos.

Existen unha serie de propiedades que son comúns a todos os elementos. Estas son:

- **Name:** Nome identificador do elemento.
- **Text:** Texto que contén o elemento.
- **Dock:** Posición do elemento relativa ó contedor. Podes situar o elemento enchendo todo o contedor, ou situalo ocupando toda a franxa esquerda, dereita, superior ou inferior, cambiando o valor desta propiedade.
- **Size.Height:** Altura do elemento.
- **Size.Width:** Ancho do elemento.

- **Location.X:** Posición do elemento na horizontal do contedor.
- **Location.Y:** Posición do elemento na vertical do contedor.
- **Anchor:** Esta opción inhabilitase ó establecer un valor para a propiedade *Dock*. A propiedade *Anchor* permite establecer que a posición do elemento sexa fixa con respecto a un ou dous dos límites do contedor no que se encontra, facendo que o redimensionar ese contedor non cambie a distancia entre o elemento e eses limites.
- **BackColor:** Con esta propiedade pódese establecer o color do fondo do elemento.

Para engadir un elemento ó formulario ou a un contedor tan só se precisa abrir o menú contextual sobre o contedor na Árbore de elementos e utilizar a opción *Add* e dentro dela elixir se se quere engadir un *Control* ou un *Container* e dentro de cada unha das opcións pódese encontrar un listado dos elementos dese tipo dispoñibles. Nada mais seleccionar un deles, este aparecerá na Árbore de elementos e poderase ver engadido na zona de traballo.

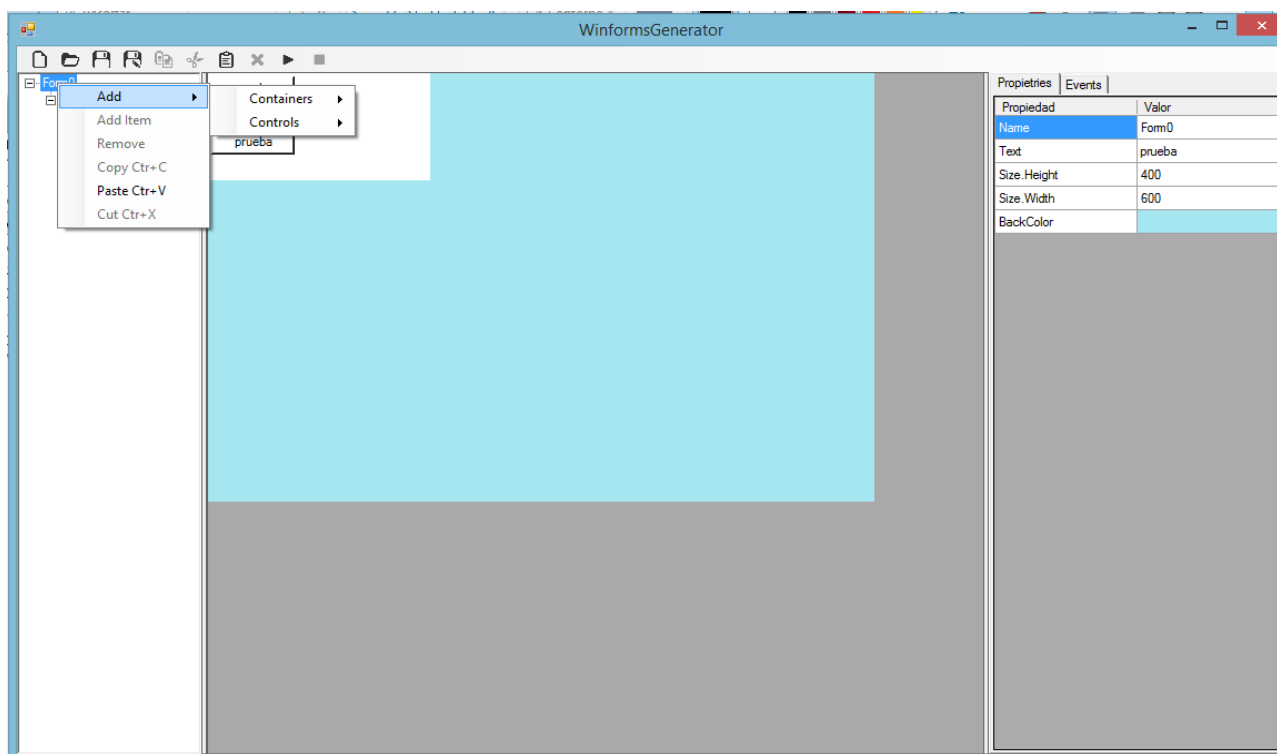


Figura 50. Engadir elemento.

Como se pode observar na Figura 50, no mesmo menú contextual pódese acceder ás opcións de copiado, cortado, pegado e borrado. As opcións de engadido e pegado tan só están

dispoñibles nos elementos contedores.

Ademais certos elementos simples, e dicir, elementos que non son contedores, posúen subelementos ou *Items* como se lles chama na aplicación. Estes elementos cando sobre eles se abre o menú contextual, a opción *Add Item* está habilitada.

As propiedades comúns ós *Items* son:

- **Name:** Nome co que se identifica o subelemento.
- **Text:** Texto que contén o interior do elemento.

Ademais a aplicación posúe unha táboa con unha serie de variables que representas os eventos que permite controlar de cada *Control*. Todos estes eventos son comúns a todos os elementos incluídos na ferramenta. Estes eventos son:

- **Click:** Este evento lánzase o facer click no elemento especificado.
- **DoubleClick:** Este evento lánzase o facer dobre click no elemento especificado.
- **Enter:** : Este evento lánzase o entrar no elemento especificado.
- **GotFocus:** Este evento lánzase cando o elemento especificado gaña o foco.
- **LostFocus:** Este evento lánzase cando o elemento perde o foco.
- **Leave:** Este evento prodúcese cando se sae dun elemento.
- **KeyDown :** Este evento prodúcese cando se presiona unha tecla cando se ten o foco no elemento especificado.
- **KeyPress:** Este evento prodúcese cando se ten presionada unha tecla cando se ten o foco no elemento especificado.
- **KeyUp:** Este evento prodúcese cando se levanta unha tecla cando se ten o foco no elemento especificado.
- **MouseClicked:** Este evento lánzase o facer click co rato no elemento especificado.
- **MouseDoubleClick:** Este evento lánzase o facer dobre click co rato no elemento especificado.
- **MouseDown:** Este evento lánzase o pulsar unha tecla do rato no elemento

especificado.

- **MouseEnter:** Este evento lánzase o facer entrar co rato no elemento especificado.
- **MouseLeave:** Este evento lánzase o saír co rato no elemento especificado.
- **MouseHover:** Este evento lánzase o parar o rato sobre o elemento especificado.
- **MouseUp:** Este evento lánzase o levantar unha tecla do no elemento especificado.
- **MouseWheel:** Este evento lánzase o utilizar a roda do rato sobre o elemento especificado.
- **Resize:** Este evento prodúcese o redimensionar o elemento especificado.

A continuación iremos falando dos diferentes elementos e as propiedades non comúns ó resto dos elementos.

3.1.3.1. Containers

Estes elementos son aqueles que teñen a característica de poder albergar a outros elementos.

Grid

Un *Grid* trátase dun contedor con forma de táboa. É un *TableLayoutPanel* na API Winforms. Permite establecer o número de filas e columnas que posúe e onde se poderá introducir un único elemento. Este elemento pode ser outro contedor e este conter mais elementos, pero cada sección do *Grid* tan só pode conter un elemento. Por defecto se engades un número de elementos maior ó numero de espacios (filas por columnas) a aplicación aumenta automaticamente o numero de filas para que os elementos teñan espacio. Se teñen espacio onde almacenarse encherán eses espacios primeiro.

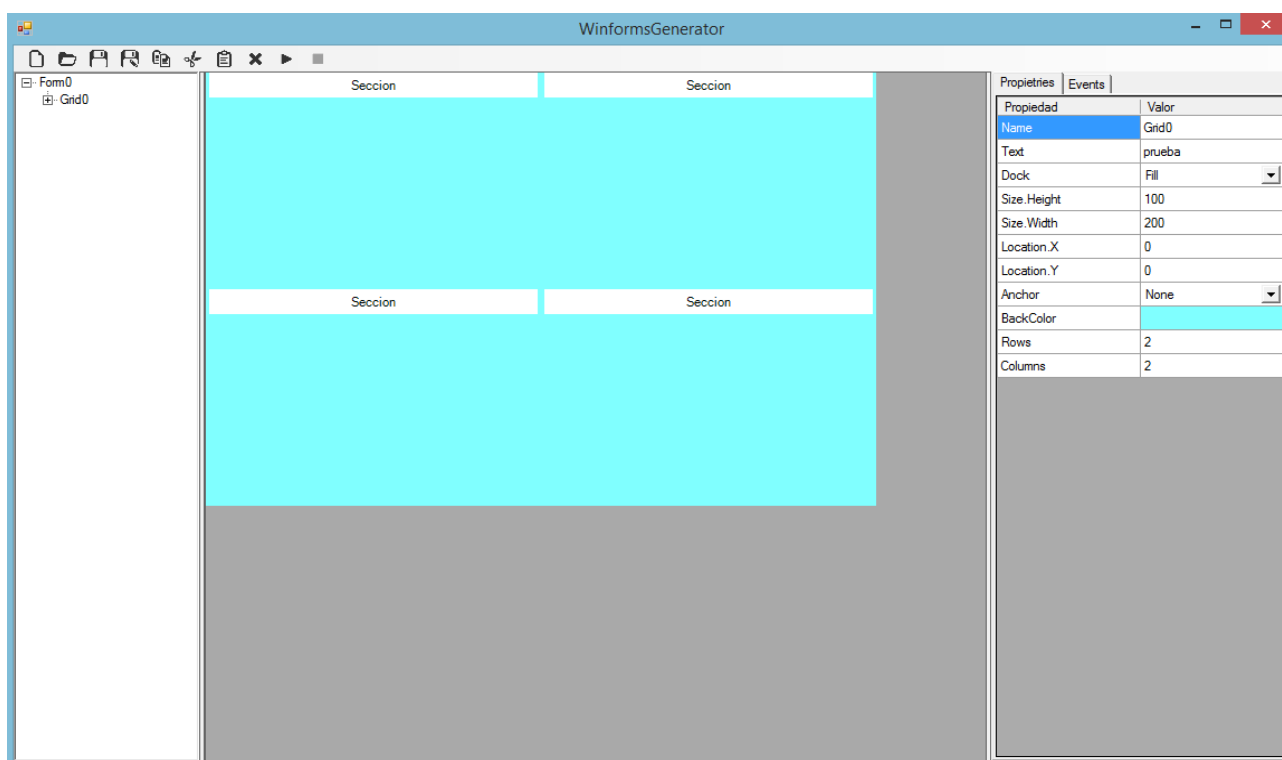


Figura 51. Grid

As súas propiedades especiais son:

- **Rows:** Especifica o número de filas que posúe o *Grid*.
- **Columns:** Especifica o número de columnas que posúe o *Grid*.

VBox

Un *VBox* é un tipo especial de *Grid* onde unicamente posúe unha única columna e un número variable de filas.

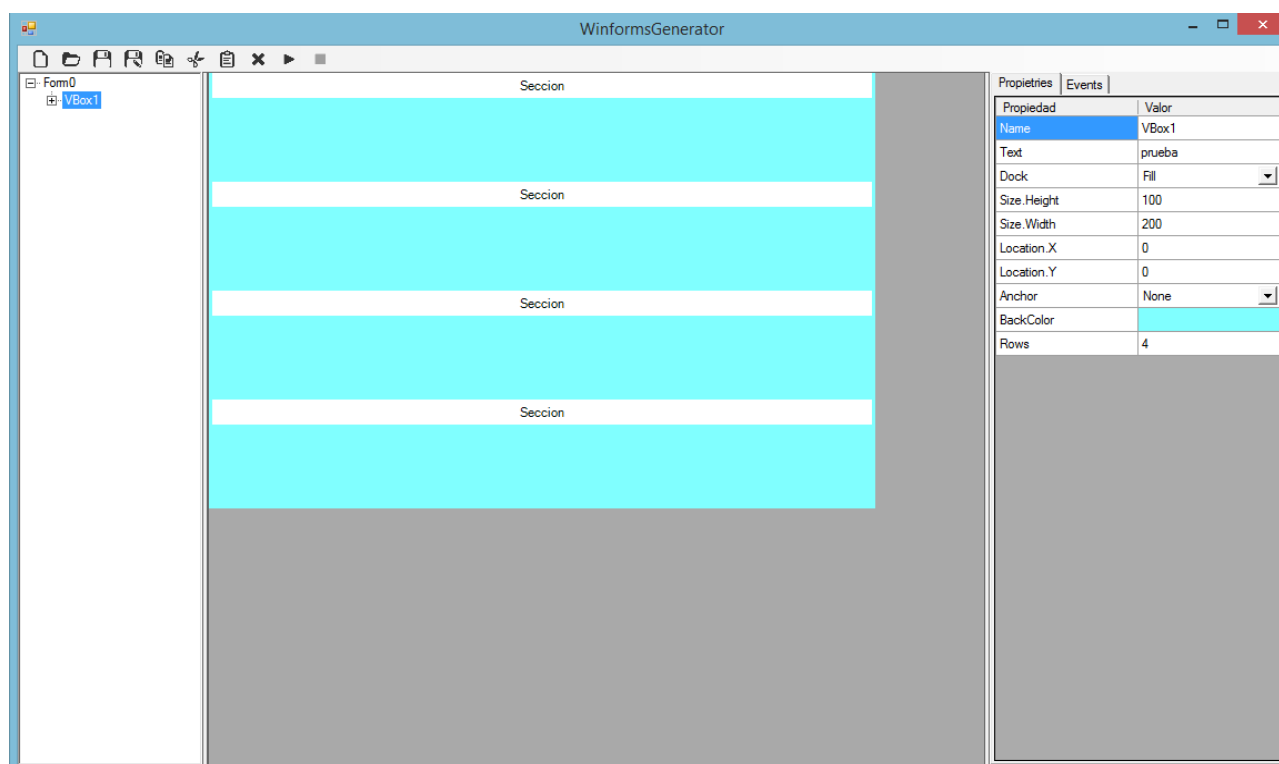


Figura 52. VBox

A sua propriedade especial é:

- **Rows:** Especifica o número de filas do contedor.

HBox

Un HBox é un tipo especial de *Grid* onde unicamente posúe unha única fila e un número variable de columnas.

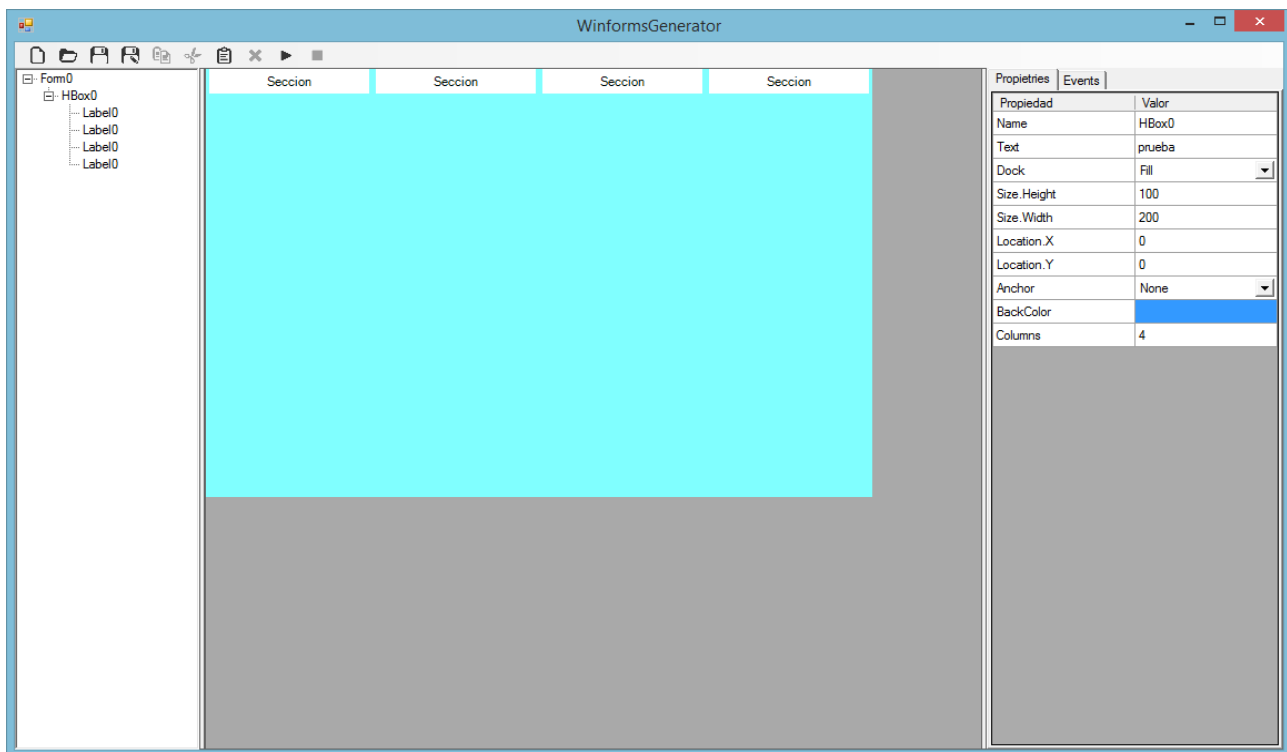


Figura 53. HBox

A sua propriedade especial é:

- **Columns:** Especifica o número de columnas do contedor.

Border

O seu equivalente na API Winforms é o *Panel*. Este é o contedor mais estándar e permite situar os elementos cubrindo a totalidade do contedor ou cubrindo a marxe superior, inferior, derecha ou esquerda.

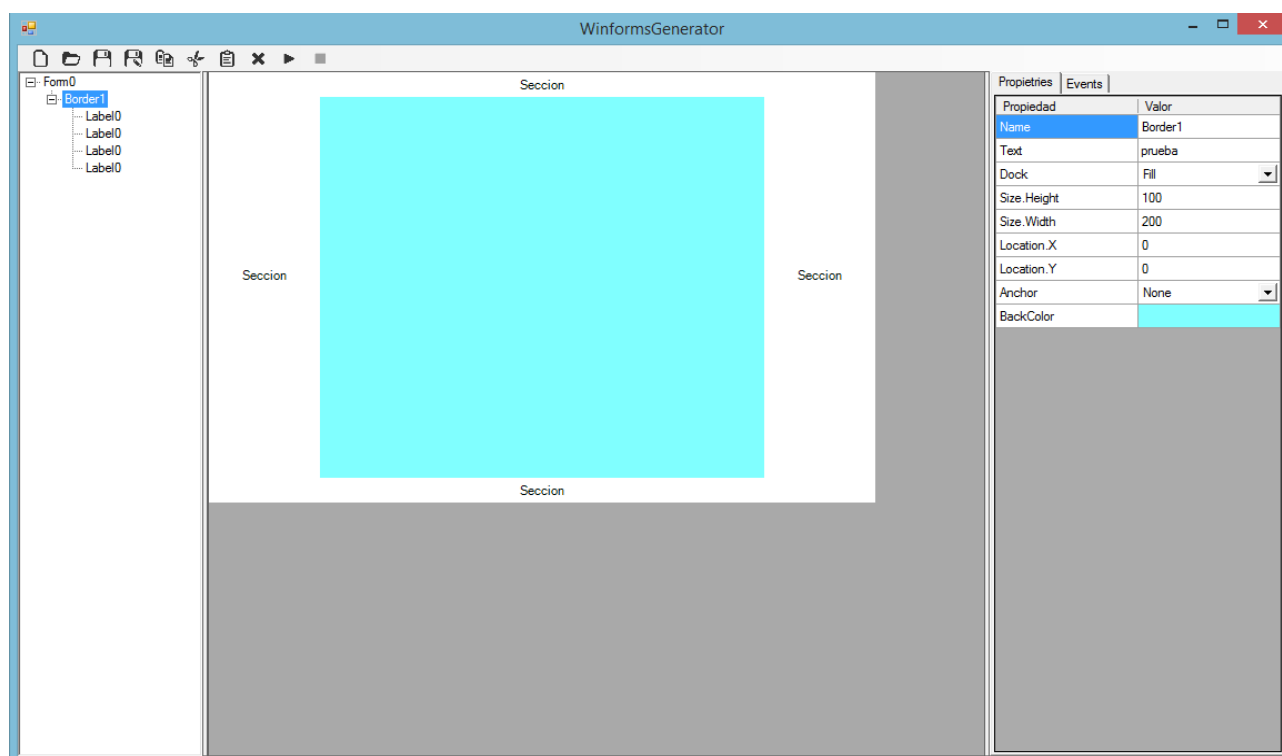


Figura 54. Border

Non posúe ningunha propiedade especial.

GroupBox

O *GroupBox* correspóndese ó elemento do mesmo nome na API Winforms. É un contedor que consiste nun cadro con título e permite situar elementos da mesma forma ca un *Border*.

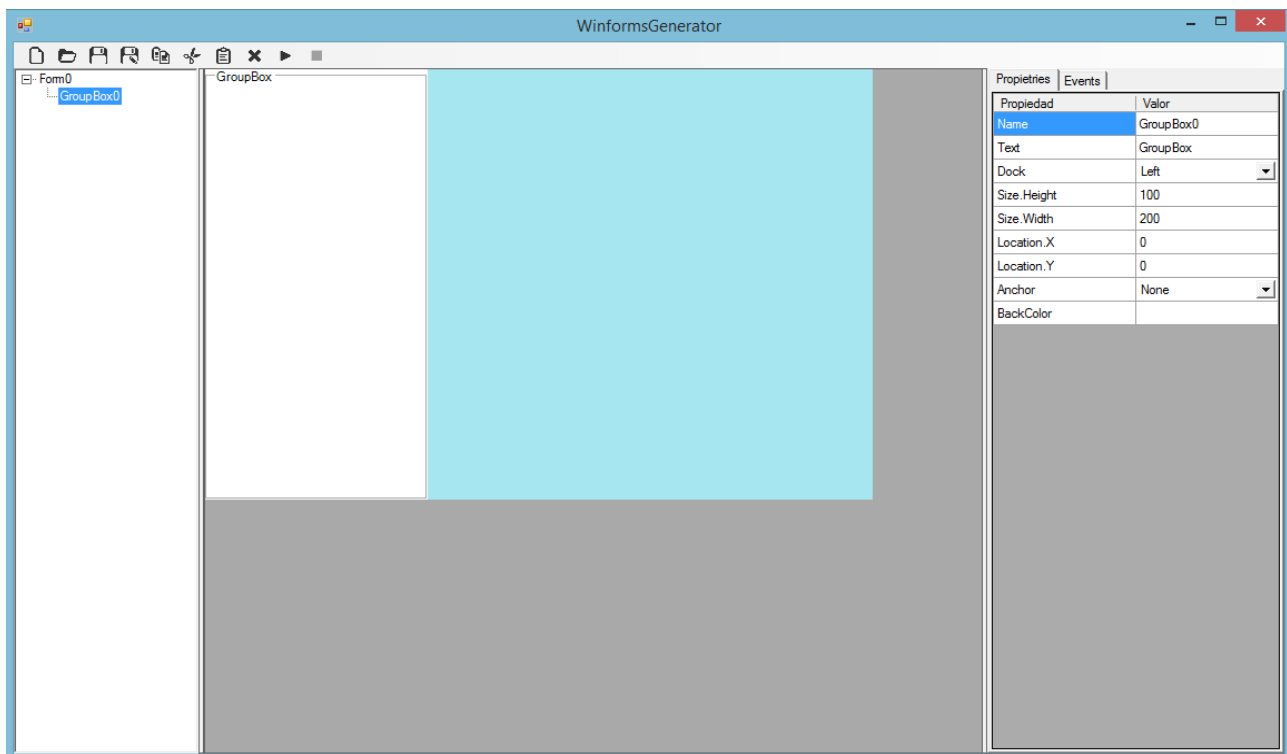


Figura 55. GroupBox

A propiedade *Text* é a que determina o texto a mostrar como título do *GroupBox*.

TabControl

Trátase dun contedor que se organiza por pestanas. Para engadir unha nova pestana basta con abrir o menú contextual sobre o elemento *TabControl*, co botón dereito do rato, e elixir a opción de engadir pestana (*Add tab*). Este elemento tan só permite engadir pestanas. Cada pestana compórtase coma se fora un *Border* e é aquí onde se poden engadir outro tipo de elementos.

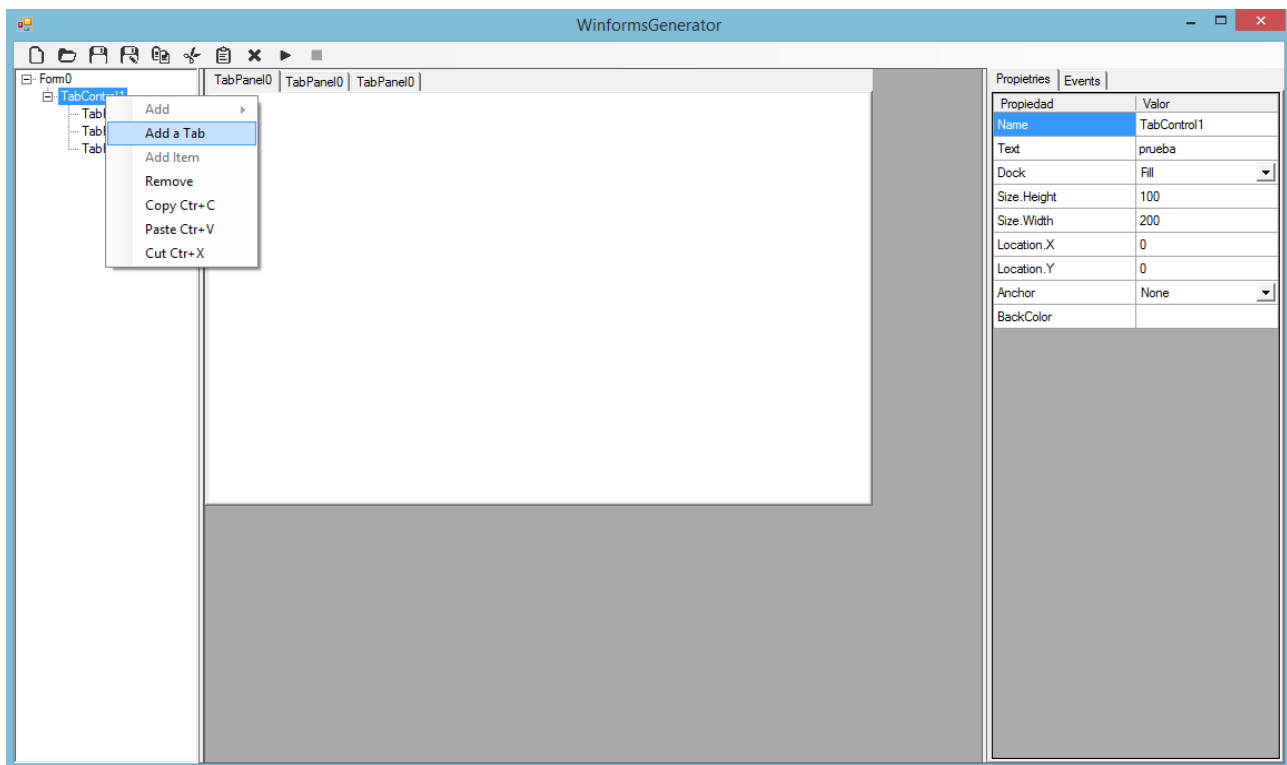


Figura 56. *TabControl*

O contenedor *TabControl* non posúe ningunha propiedade especial ó igual que as pestanas que o forman. O título destas pestanas correspóndese coa propiedade *Text*.

3.1.3.2. Controls

Os *Controls* trátanse de obxectos simples de interface. Cando nos referimos a obxectos simples quero dicir que son elementos que non conteñen outros elementos. Estes elementos son os que realmente lle dan características de funcionamento á interface. Todos os elementos desta categoría posúen o mesmo nome que o seu equivalente na API Winforms polo que non é preciso explicalos.

Button

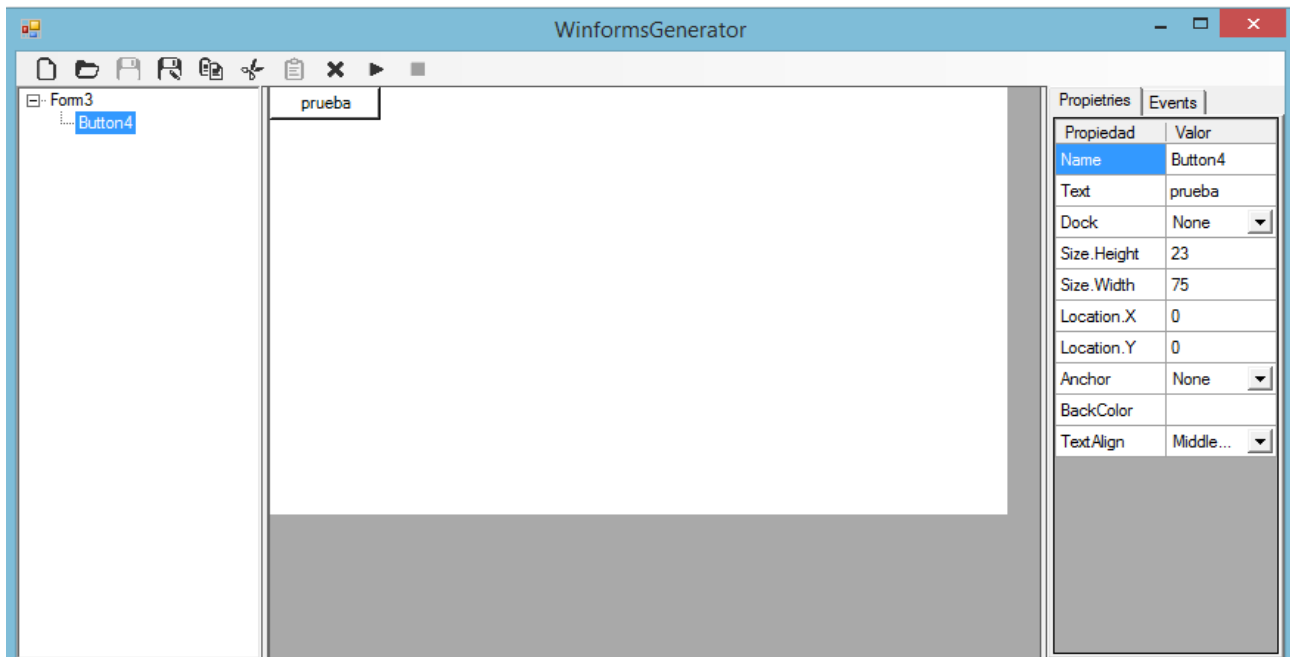


Figura 57. Button

As propiedades que posúe en diferencia ós outros elementos son:

- **TextAlign:** Esta propiedade establece a aliñación do texto dentro do botón.

Label

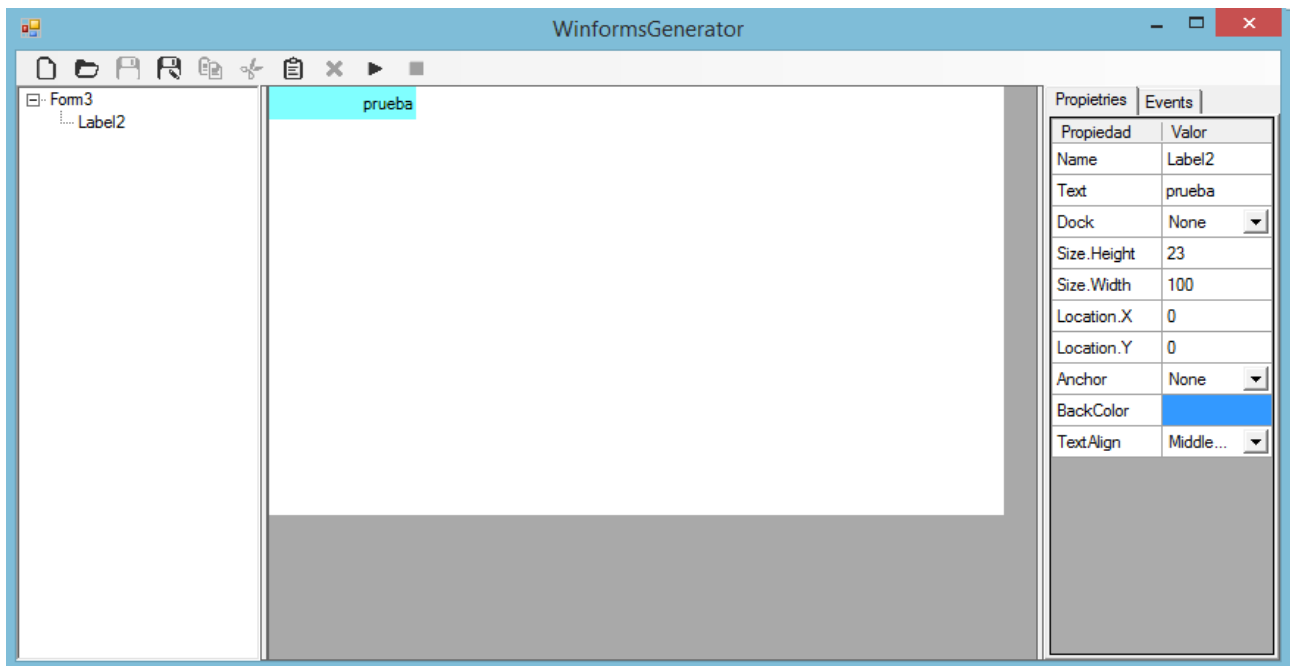


Figura 58. Label

As propiedades que posúe en diferencia ós outros elementos son:

- **TextAlign:** Esta propiedade establece a aliñación do texto dentro do cadro de texto.

TextBox

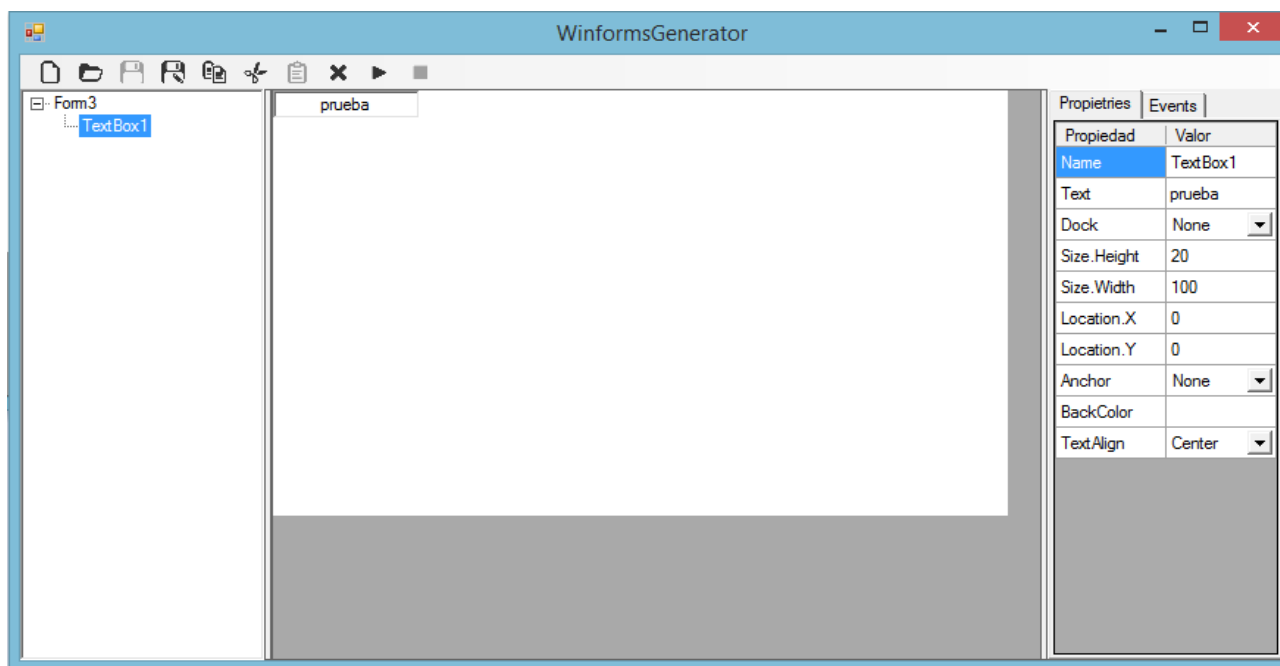


Figura 59. TextBox

As propiedades que posúe en diferencia ós outros elementos son:

- **TextAlign:** Esta propiedade establece a aliñación do texto dentro do cadro de inserción de texto.

RadioButton

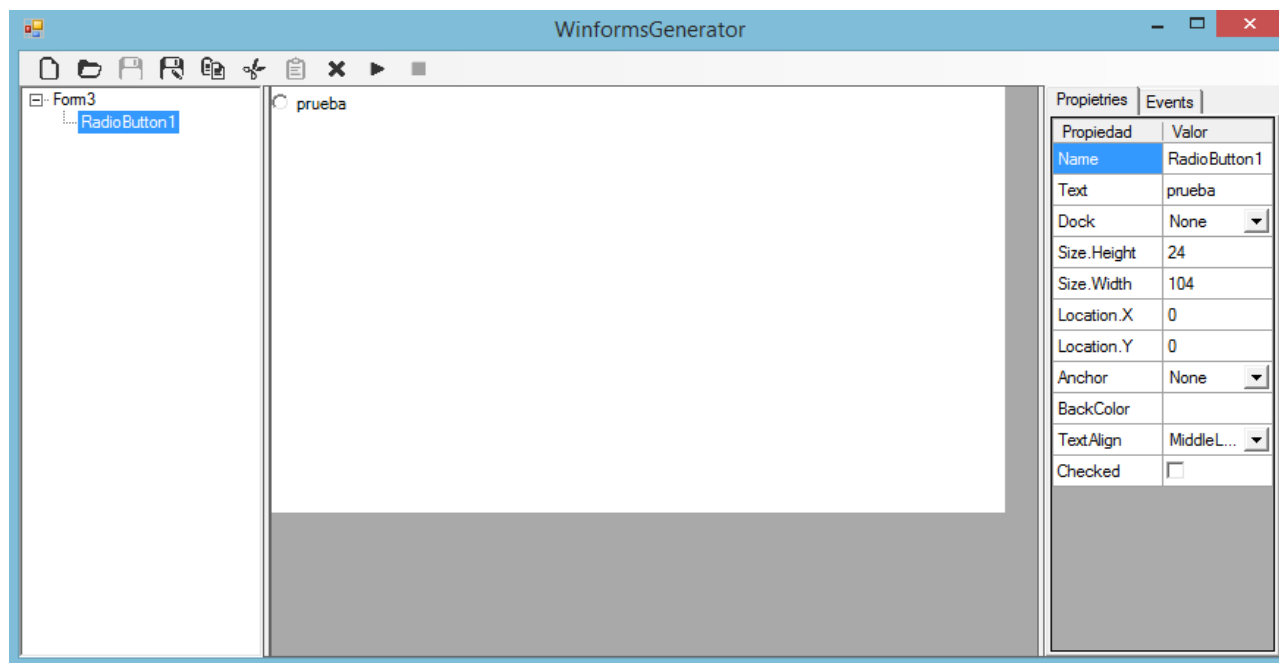


Figura 60. RadioButton

As propiedades que posúe en diferencia ós outros elementos son:

- **TextAlign:** Esta propiedade establece a aliñación do texto do elemento.
- **Checked:** Determina se o elemento esta seleccionado ou non por defecto.

CheckBox

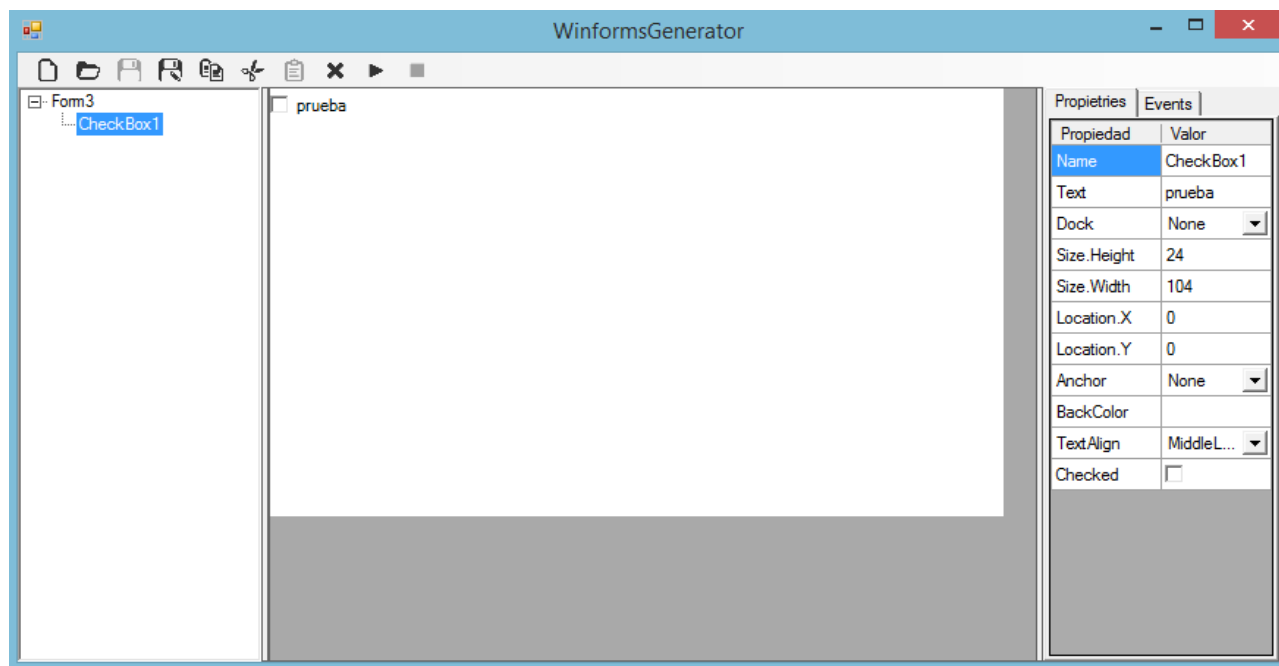


Figura 61. CheckBox

As propiedades que posúe en diferencia ós outros elementos son:

- **TextAlign:** Esta propiedade establece a aliñación do texto dentro do elemento.
- **Checked:** Determina se o elemento está seleccionado por defecto ou non.

ProgressBar

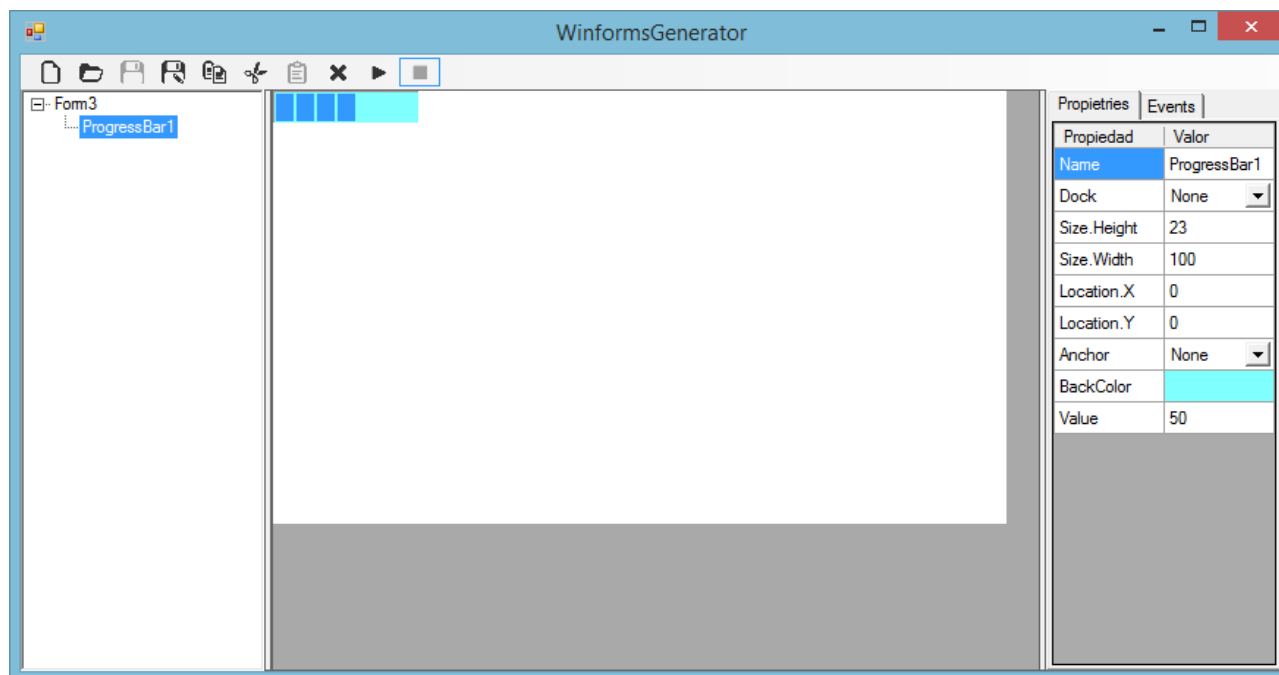


Figura 62. ProqressBar

As propiedades que posúe en diferencia ós outros elementos son:

- **Value:** Establece a porcentaxe de progreso que marca o barra de progreso por defecto.

DateTimePicker

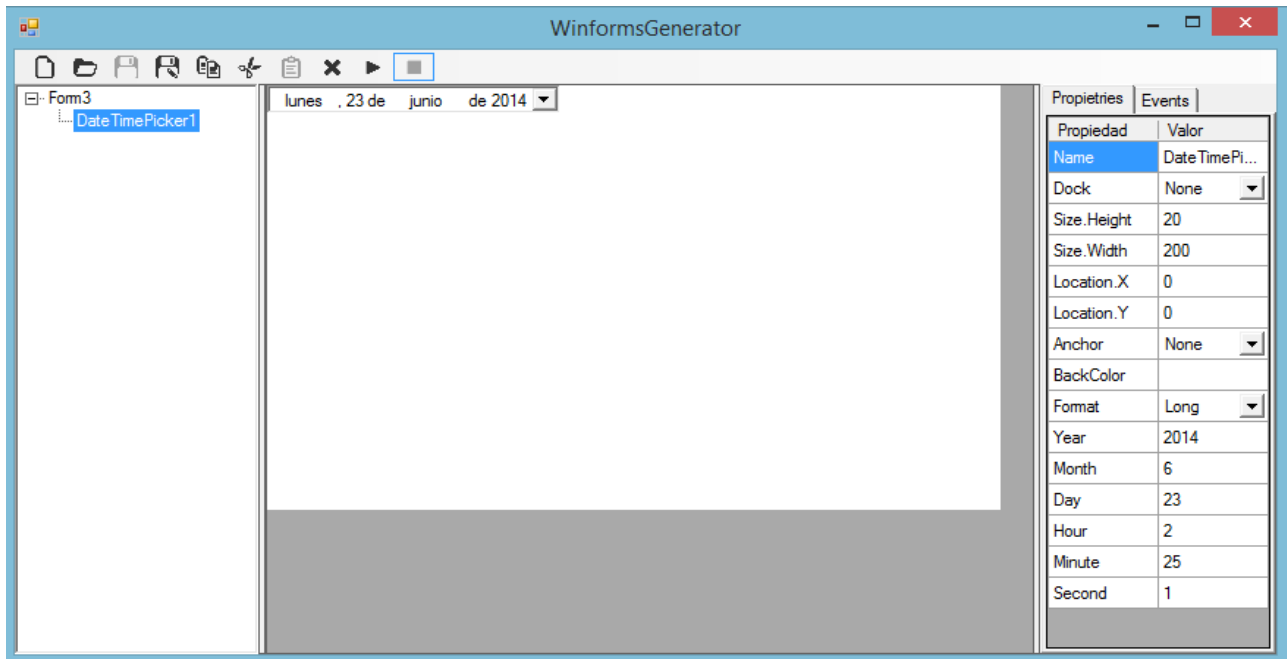


Figura 63. DateTimePicker

As propiedades que posúe en diferencia ós outros elementos son:

- **Format:** Establece el formato en que se mostra la fecha y hora de este elemento
- **Year:** Ano da data que marca por defecto.
- **Month:** Mes da data que marca por defecto.
- **Day:** Día da data que marca por defecto.
- **Hour:** Horas que marca por defecto no reloxio do elemento.
- **Minute:** Minutos que marca por defecto no reloxio do elemento.
- **Second:** Segundos que marca por defecto o reloxio do elemento.

MonthCalendar

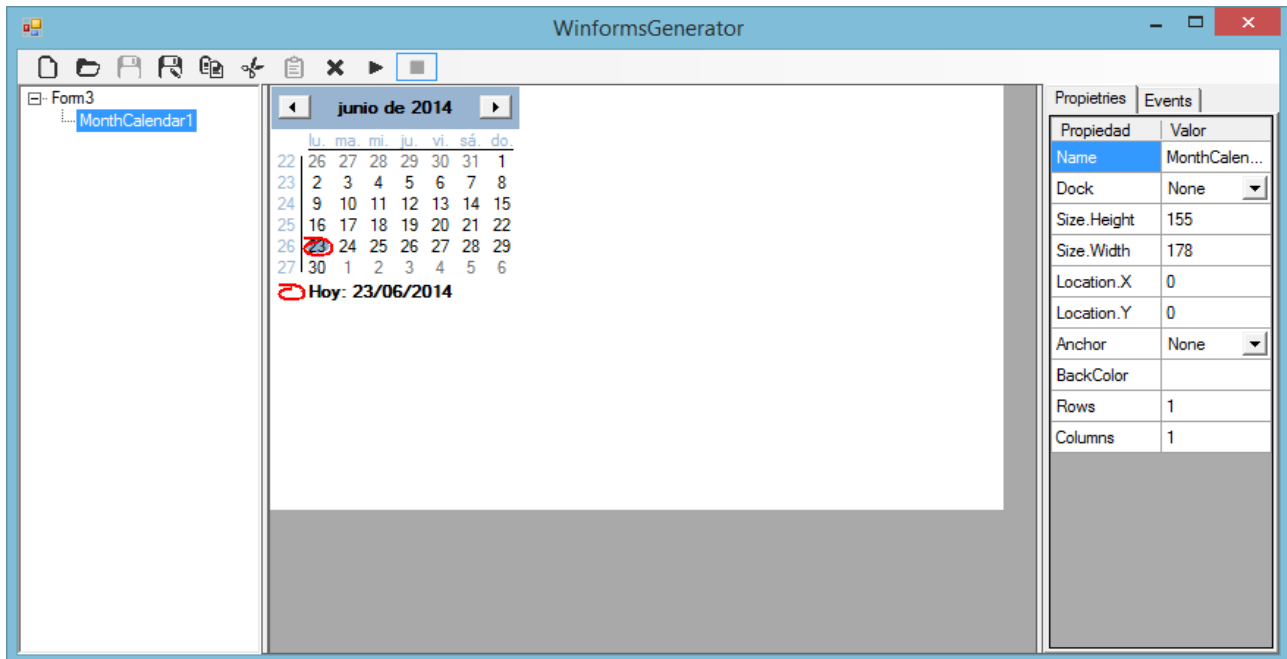


Figura 64. MonthCalendar

As propiedades propias de este elemento son:

- **Rows:** Número de filas que terá o calendario creado.
- **Columns:** Número de columnas que terá o calendario.

Splitter

Este elemento permite crear separadores entre elementos que conforman a interface en desenvolvemento. Ten como aspecto especial que a propiedade *Dock* non pode tomar o valor *Fill*.

PictureBox

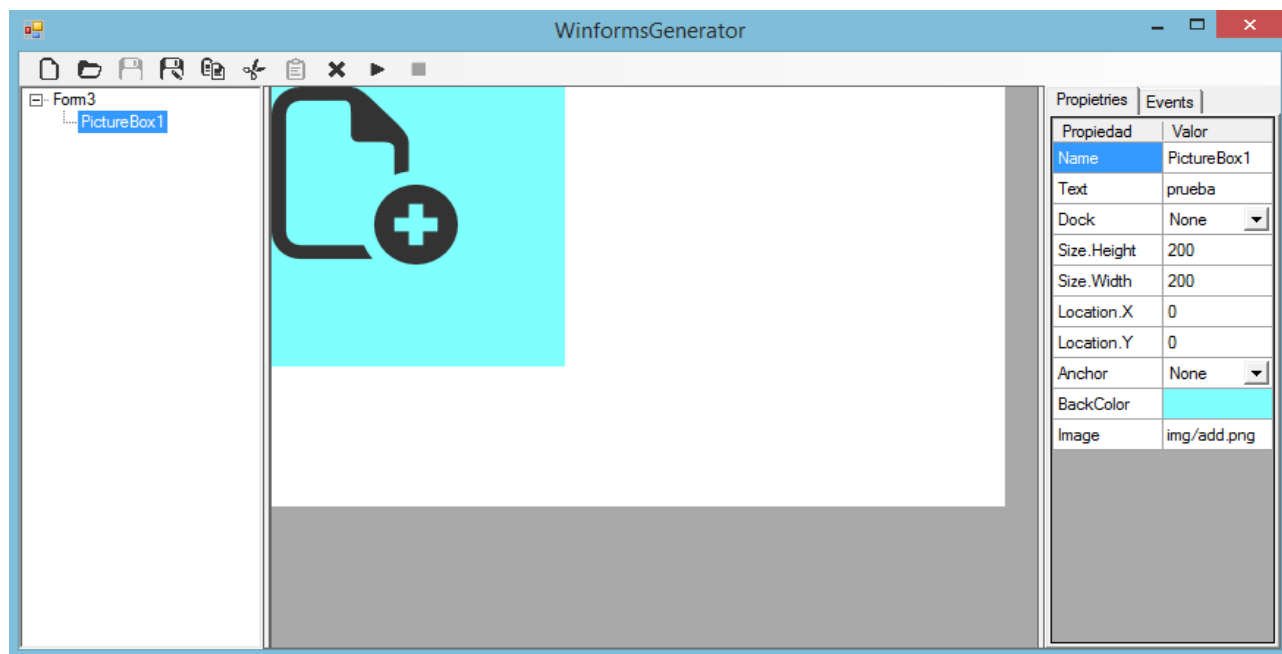


Figura 65. PictureBox

As propiedades que caracterizan a este elemento son:

- **Image:** Trátase dunha *String* que se debe encher coa ruta da imaxe que se quere mostrar.

ComboBox

O comboBox é un tipo especial de *Control* xa que a pesar de ser un elemento sinxelo contén unha serie de elementos moi concretos. Neste elemento os subelementos ou *Items* coma se lles chama na aplicación, son unicamente *Strings*. Estes elementos engádense a través do menú contextual e coa opción *Add Item*. Esta opción tan só esta habilitada nos controles que soportan subelementos ou *Items*.

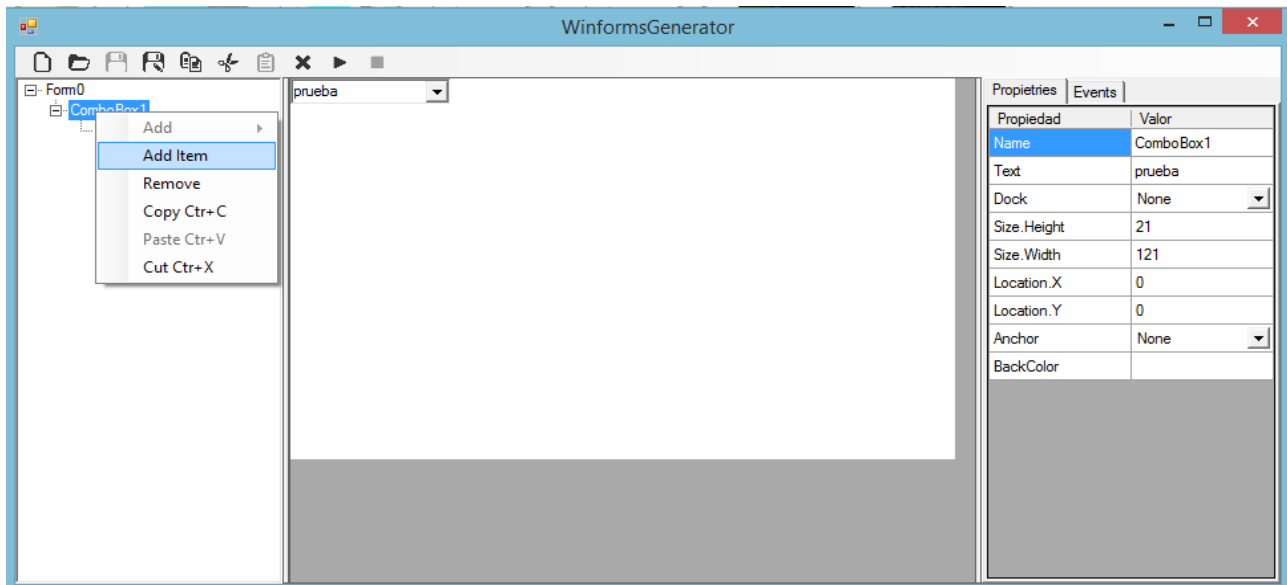


Figura 66. ComboBox

MenuStrip

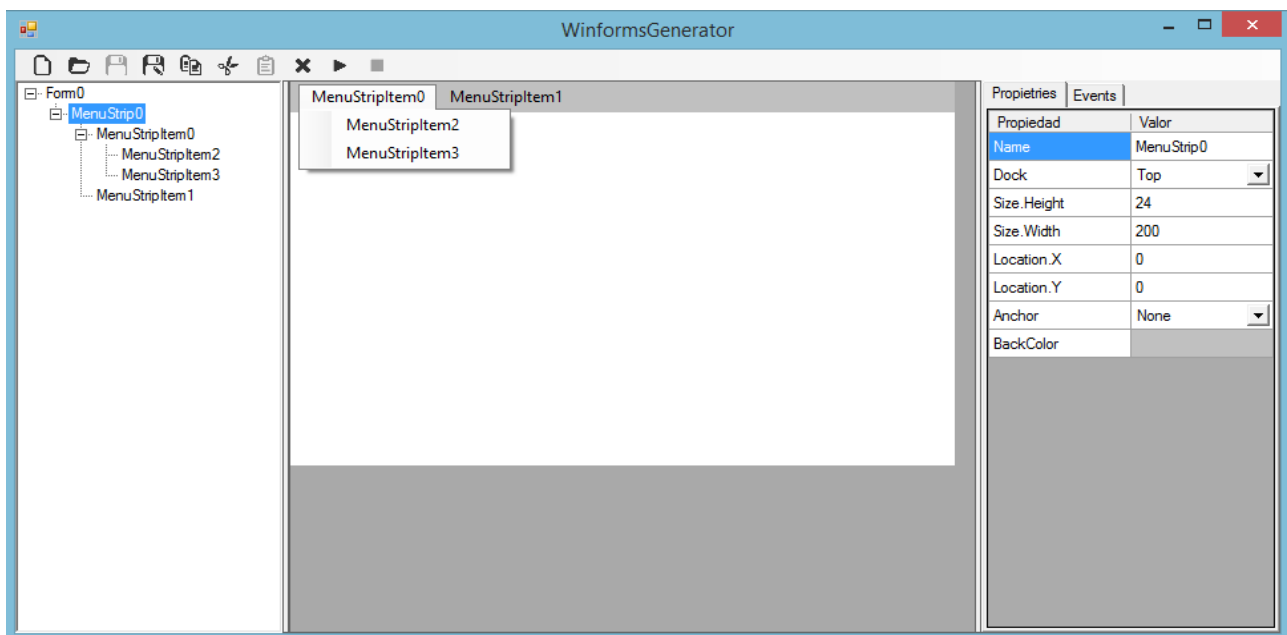


Figura 67. MenuStrip

Nesta ocasión os *Items* son o equivalente na API Winforms *ToolStripMenuItem*. Ademais neste caso os *Items* que posúe o *MenuStrip* tamén poden insertar mais *subItems* dentro dos seus *Items*. Para elo, no menú contextual dos *Items* aparece habilitada a opción *Add Item*.

TreeView

O *treeView* posúe, ó igual que o elemento anterior, *Items* e *subItems*, Ningún destes *Items*

posúe propiedades diferentes ás xerais de todos os *Items*. Estes subelementos correspóndense cos *TreeNode*s da API Winforms.

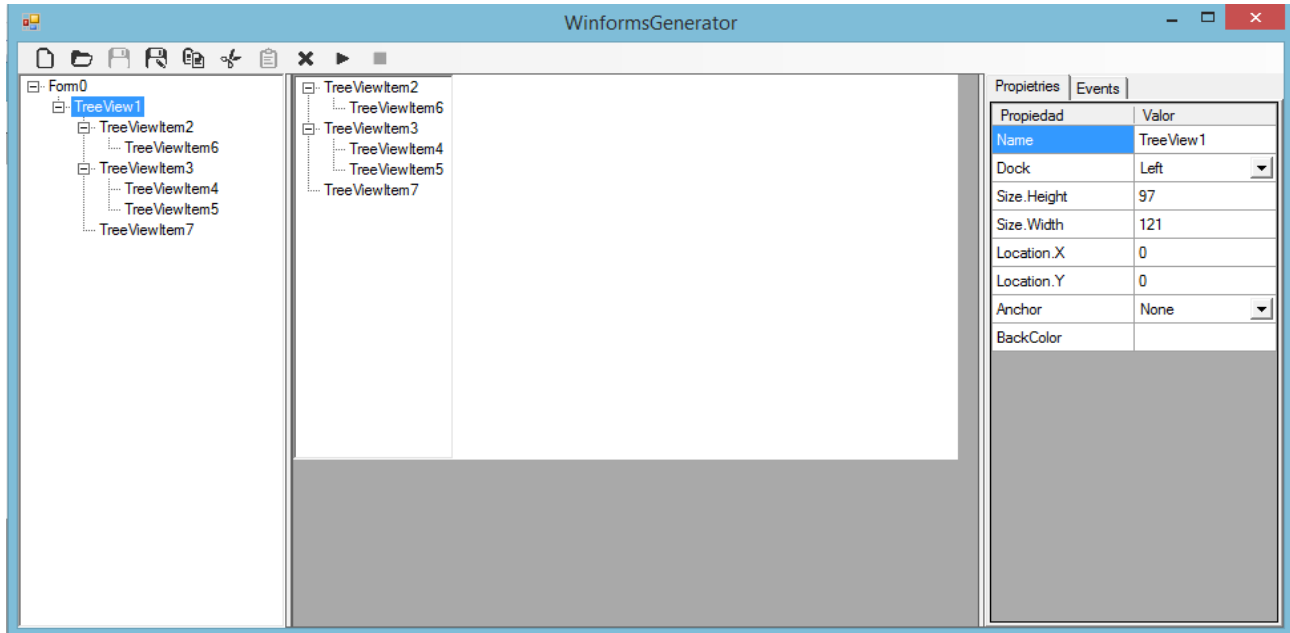


Figura 68. TreeView

ToolBar

A barra de ferramentas ou *ToolBar* contén unicamente *Items* e non *SubItems*. Estes *Items* posúen unicamente as propiedades *Name* e *Text* comúns a todos os *Items*. Estes *Items* que se lle poden engadir a este elemento son os *ToolBarButton* da API Winforms.

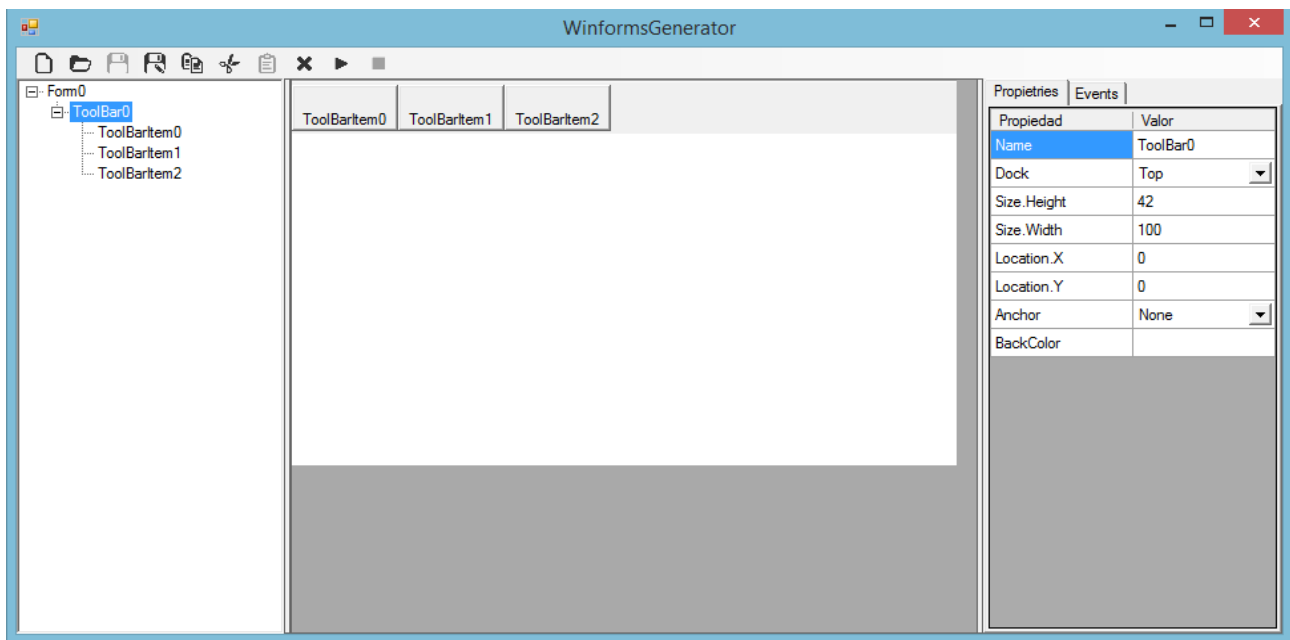


Figura 69. ToolBar

StatusBar

A *StatusBar* tiene solamente *Items*. Estos poseen las siguientes características diferentes a los comunes:

- **Width:** Establece la longitud de los *Items* que contienen a la *ToolBar*.
- **BorderStyle:** Define el estilo del borde de los *Items* de la *ToolBar*.

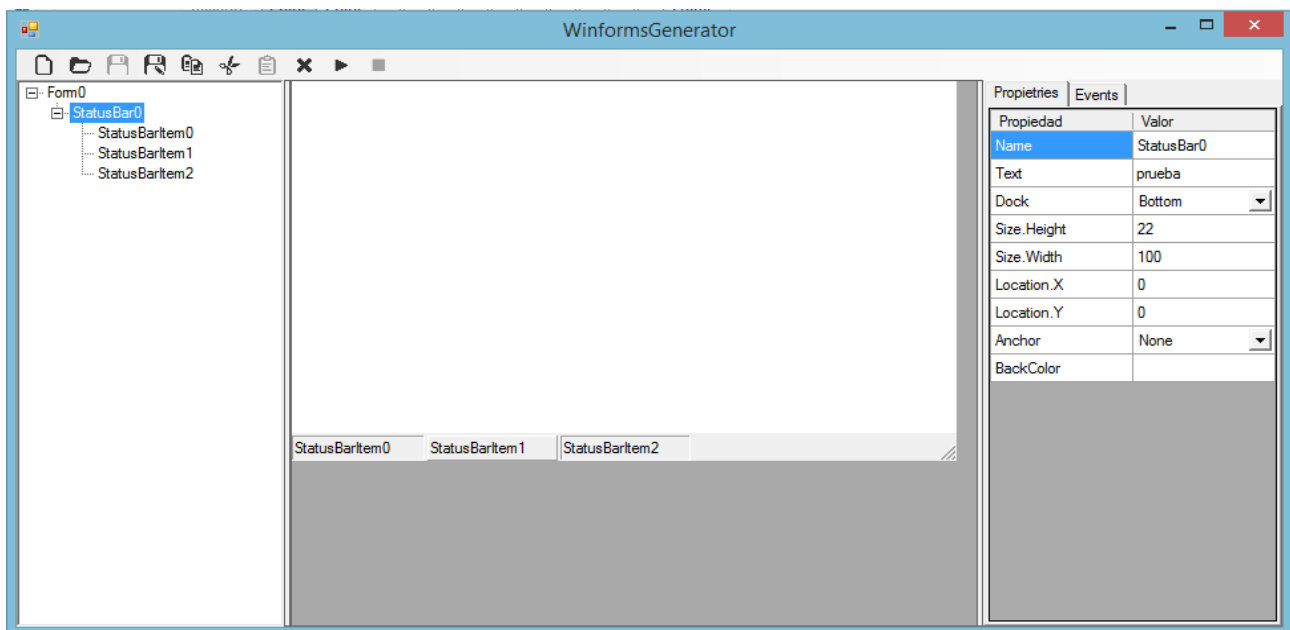


Figura 70. StatusBar

ListView

O *ListView* ten a posibilidade de engadírselle *Items* que na API Winforms son os *ListViewItem*.

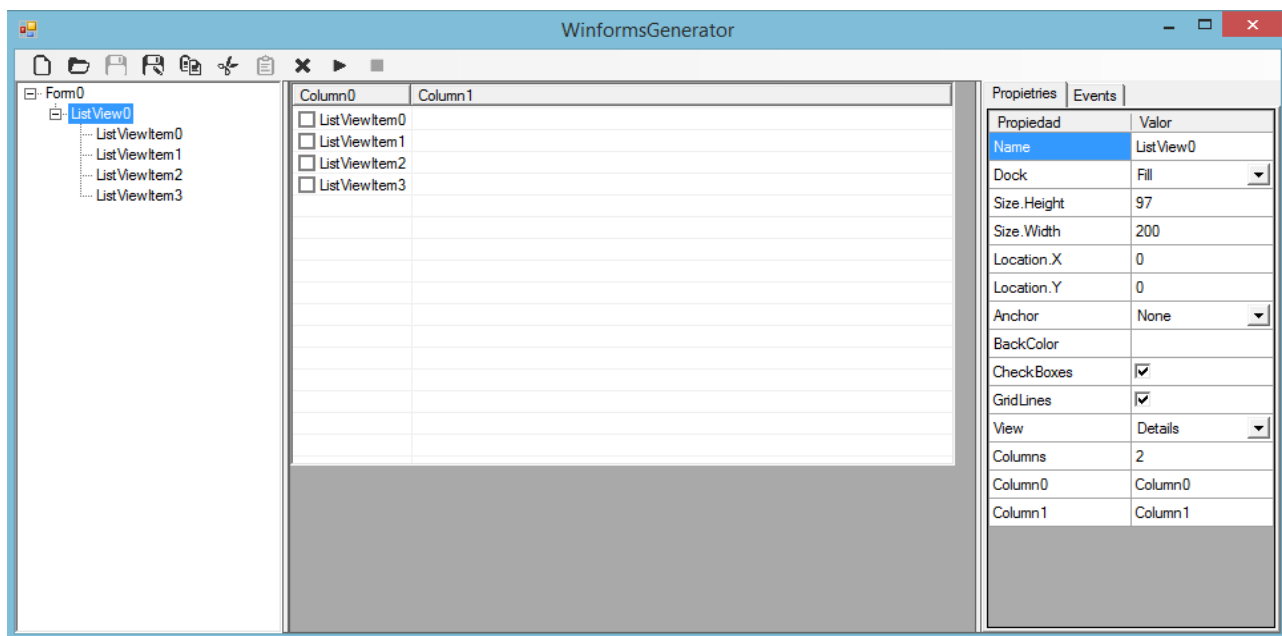


Figura 71. *ListView*

Ademais este elemento posúe unha serie de características que non comparte cos outros elementos. Estas son:

- **CheckBoxes:** Con esta característica se define se no *ListView* o lado de cada *Item* aparece un cadro de selección.
- **GridLines:** Esta característica establece se o *ListView* posuirá ou non liñas separadoras a modo de táboa.
- **View:** Esta propiedade controla o aspecto do *ListView*.
- **Columns:** Neste último campo establécese o número de columnas que posuirá o *ListView*. O cambiar este valor, a táboa de propiedades aumenta con tantos campos coma columnas se lle estableceran. Estes novos campos controlan o título de cada columna.

DataGridView

Por último o *DataGridView*. Este posúe *Items* que non son mais que os *Strings* que aparecerán nas celdas de este elemento.

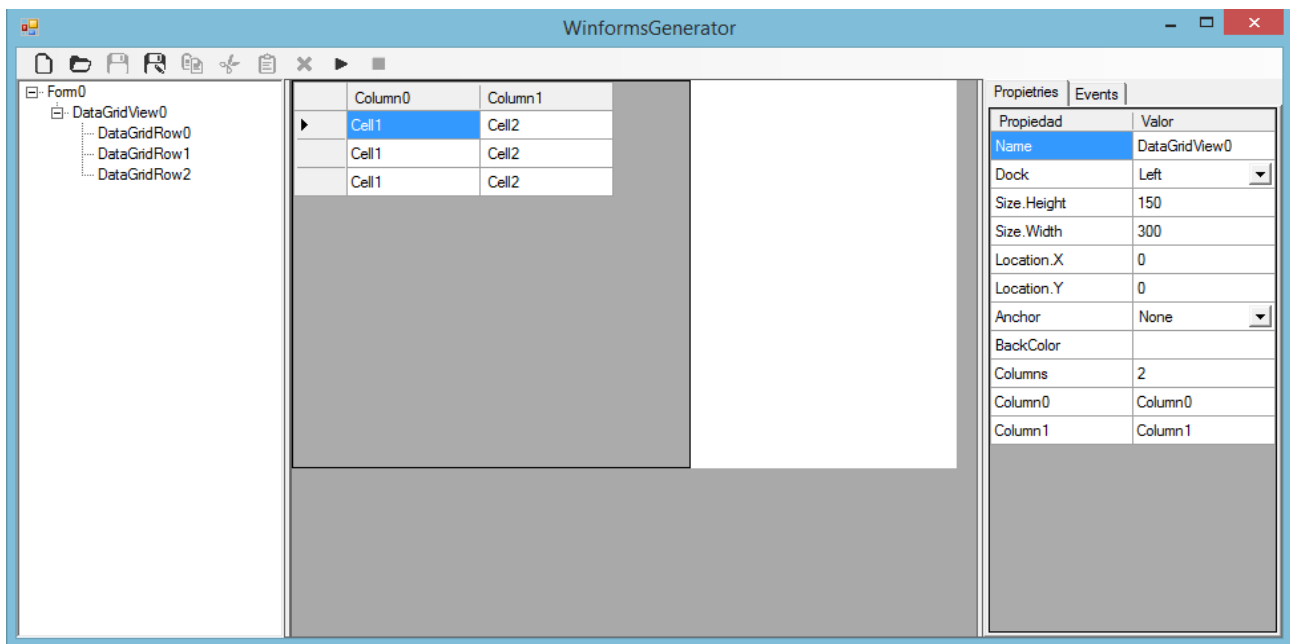


Figura 72. *DataGridView*

O *DataGridView* posue as seguintes características propias:

- **Columns:** Esta propiedade establece o número de columnas do *DataGridView*. Ó cambiar o valor desta propiedade, a táboa de propiedades aumenta para permitir establecer os títulos das columnas.

Nos *Items* que compoñen o *DataGridView* existe unha propiedade tamén que non comparte co resto de *Items*:

- **Values:** Os valores, separados por comas, introducidos nesta fila, establecen el texto que aparecerá nas celdas que formas a fila correspondente.

3.2. Exportar e importar interfaces

O obxectivo principal da aplicación é server de axuda a un desarrollador de aplicacións en C#, para que con esta ferramenta sexa capaz de crear unha interface gráfica para o seu proxecto, dunha forma rápida e sinxela.

3.2.1. Exportar

Unha vez deseñada a interface que queremos utilizar no noso propio proxecto, debemos exportala a un arquivo Xml. Para isto utilizaremos función *Save* ou *Save As*. Unha vez feito isto xa temos a nosa interface gardada e dispoñible para engadir ó noso proxecto.

3.2.2 Importar

Para utilizar a nosa interface xa deseñada e exportada, precisamos primeiramente crear unha nova solución. Unha vez creada, precisamos importar a biblioteca *WinformsImport* ó noso directorio de *Referencias*, coma na Figura 73. Esta Librería a poderemos encontrar no directorio “Aplicación final”.

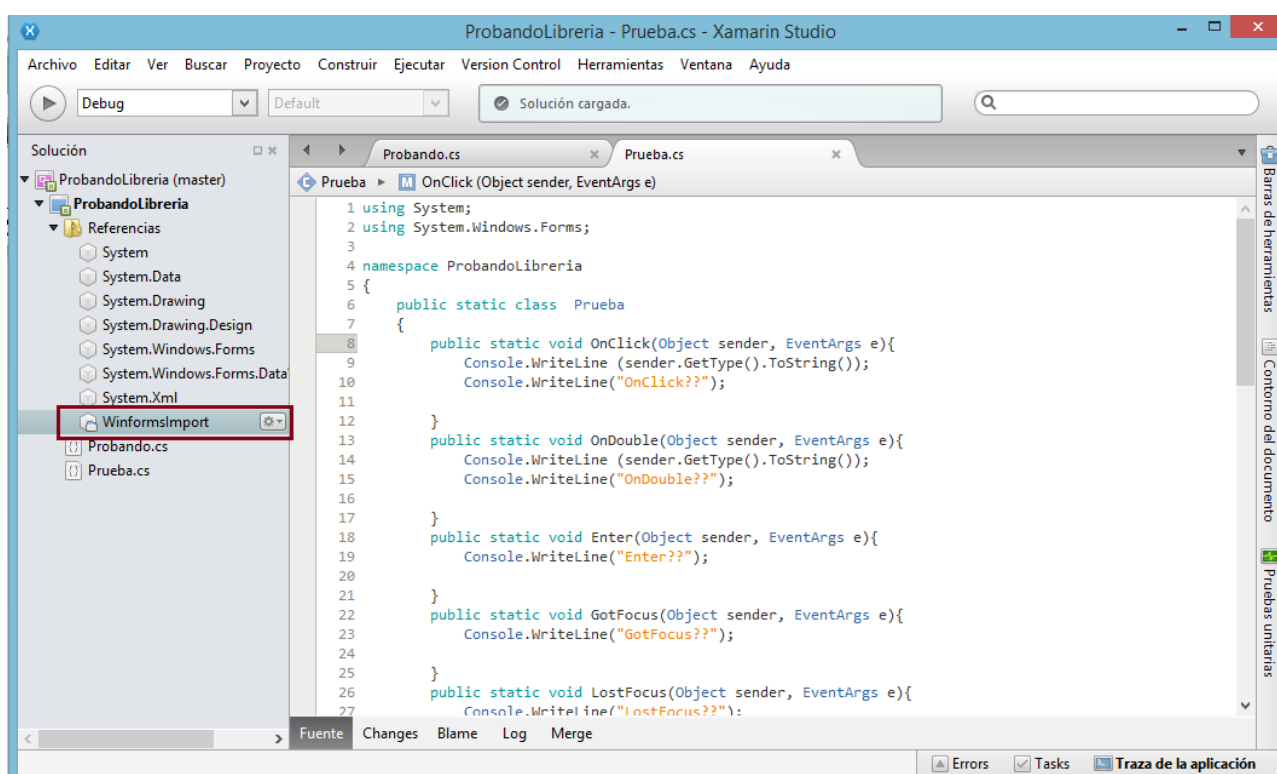


Figura 73. Engadindo Referencias

Unha vez feito isto xa podemos utilizar a función de importación. A función é *OpenWinform(String XmlFile)*. Executándoa coma na figura 74 obteremos o formulario desexado.

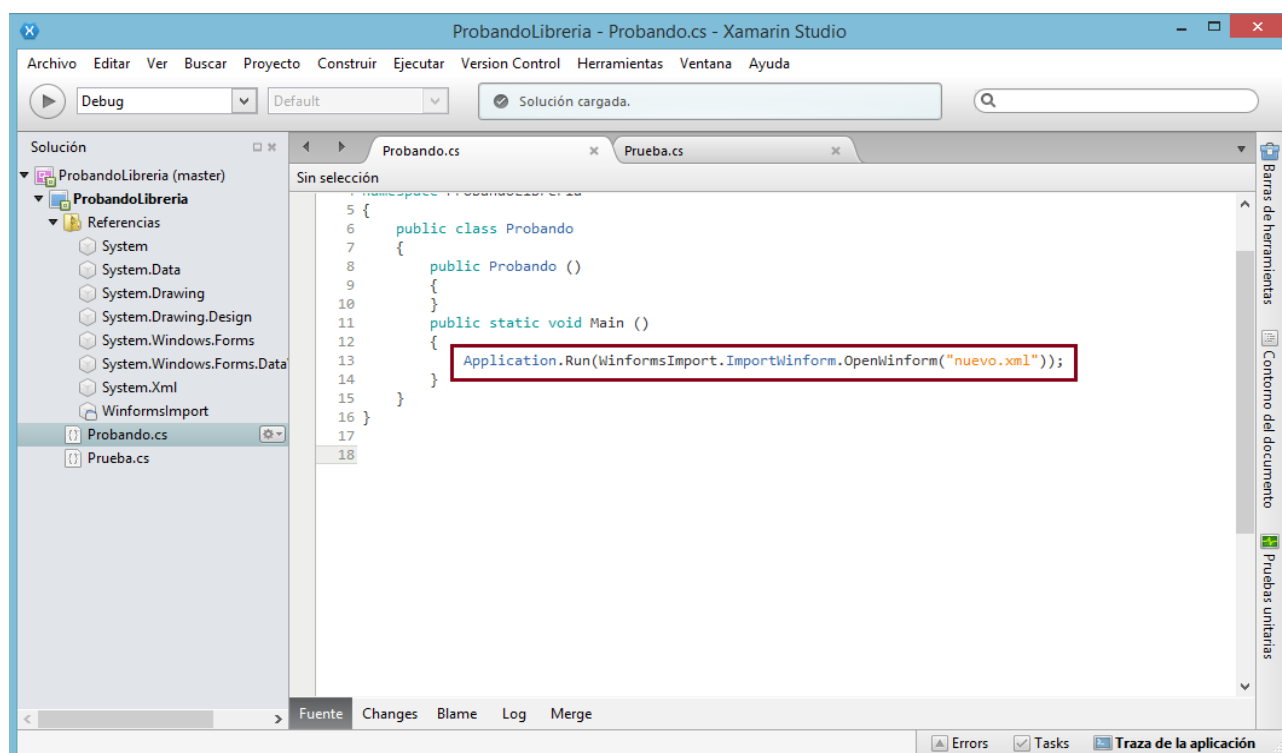


Figura 74. Como utilizar unha interface creada

Para a configuración de eventos, na aplicación de deseño hai que especificar no campo do evento desexado, do elemento preciso, o nome dunha función estática que logo deberemos implementar no noso proxecto. A librería a encontrará automaticamente. O formato da función deberá ser:

public static void Funcion(object sender, EventArgs e)

É moi importante non esquecer dos argumentos especificados, sen eles non funcionará correctamente.

4. Posibles erros

4.1. Non ter instalado o Software preciso

Se a hora de executar a aplicación nun sistema GNU/Linux, esta non ten no menú contextual a opción de abrir con *Mono Runtime*, o mais seguro e que non o teñas instalado.

Repasa a sección de instalación de este manual.

Se cando queres facer a clonación do programa, non o encontras a aplicación *Git* a través dos directorios de Windows, ou a executala en terminal, en calquera dos sistemas soportados, non encontra o comando comproba se está instalado. Para mais información acerca do software preciso volta á sección de instalación.

4.2. Erro ó abrir ou importar unha interface

Se cando se quere abrir un arquivo Xml que contén, supostamente unha interface feita coa aplicación, ou se se pretende importalo utilizando a biblioteca proporcionada, e sae un error coma o da Figura 75, é porque se seleccionou un arquivo Xml non válido.

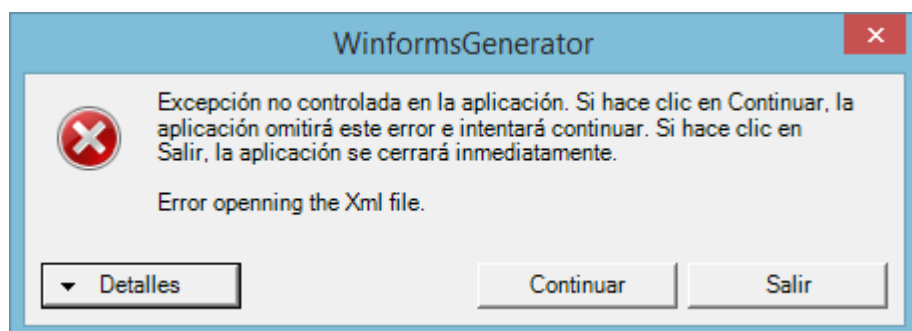


Figura 75. Erro o abrir unha interface

