
RESUMEN

Bla, bla, bla ...

Agradecimientos

Bla, Bla, Bla ...

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación	1
1.3. Objetivos	2
1.4. Estructura de la memoria	3
2. Estado del Arte	5
2.1. Análisis de requerimientos	5
2.2. Herramientas actuales	6
2.2.1. Surveyor	6
2.2.2. LimeSurvey	7
2.2.3. Survey Project	7
2.2.4. Google Forms	7
2.2.5. Comparativa	7
3. Contexto Tecnológico	11
3.1. Modelo-vista-controlador	11
3.2. Flask	12
3.3. Python	13
3.4. Jinja2	13
3.5. Otros lenguajes	14
3.6. Bootstrap	14
3.7. SQLAlchemy	15
3.8. WTForms	15
3.9. GitHub	15
3.10. Software de apoyo	16

ÍNDICE GENERAL

4. Esqueleto de la aplicación construida	17
4.1. Flask	17
4.2. Modelo Vista Controlador implantado	20
4.3. Plataforma Modular	20
5. Enrutamiento	23
5.1. Resolución de rutas en Flask	23
5.2. Rutas de la aplicación	24
5.2.1. Módulo main	24
5.2.2. Módulo de autenticación	24
5.2.3. Módulo de investigador	25
5.2.4. Módulo del encuestado	25
5.2.5. Módulo de feedback	26
5.2.6. Módulo de visualización de estadísticas	26
6. Seguridad	27
6.1. Autenticación de usuarios	27
6.2. Control de acceso	28
7. Controlador	31
7.1. Estructura.	31
7.2. Formularios	32
7.2.1. Creando un formulario	32
7.2.2. Creando un formulario dinámico	33
7.2.3. Creando un campo personalizado	33
7.2.4. Creando un validador personalizado	33
7.2.5. Trabajo con formularios	35
8. Plantillas	37
8.1. Motor de plantillas Jinja2	37
8.2. Estructura	38
8.3. Traducción	40
8.3.1. Selección de idioma	40
8.3.2. Traducción de ficheros	40
8.4. JavaScripts	42

9. Pruebas	43
9.1. Tests unitarios y cobertura de código	43
9.2. Pruebas de carga	44
9.3. Pruebas de Sistema	44
10. Gestión del proyecto	45
10.1. Ciclo de vida del desarrollo.	45
10.2. Gestión del tiempo	45
11. Conclusiones	49
11.1. Conclusiones generales	49
11.2. Trabajo futuro	49
11.3. Conclusión personal	51
A. Manual de administrador	53
A.1. Consideraciones	53
A.2. Requisitos hardware	53
A.3. Requisitos software	54
A.4. Instalación	54
A.5. Configuración	55
A.6. Base de datos	56
A.6.1. Configuración de la base de datos a usar	56
A.6.2. Creación de la base de datos	56
A.7. Inicio del programa	57
A.8. Asignación del rol investigador a un usuario	57
A.9. Actualización	57
A.10. Log	58
B. Manual de investigador	59
B.1. Inicio de sesión	59
B.1.1. OpenID	59
B.1.2. Correo	60
B.2. Creación de encuestas	62
B.2.1. Crear o editar una encuesta	62
B.2.2. Añadir y editar consentimientos	67
B.2.3. Insertar una nueva sección	67
B.2.4. Añadir nuevas preguntas	70

ÍNDICE GENERAL

B.2.4.1. Preguntas cuya respuesta es Si o No	70
B.2.4.2. Preguntas cuya respuesta es un texto	73
B.2.4.3. Preguntas cuya respuesta es una opción posible	73
B.2.4.4. Preguntas cuya respuesta es una escala likert	76
B.2.4.5. Preguntas dependientes de la respuesta a otra pregunta	76
B.2.4.6. Preguntas de Juegos	79
B.2.5. Editar y eliminar preguntas	82
C. Guía de desarrollo	85
C.1. Preguntas de tipo fecha	85
C.1.1. Modificación de la base de datos	85
C.1.2. Modificación del módulo Researcher	86
C.1.3. Modificación del módulo Surveys	87
C.2. Preguntas de tipo casillas de verificación	88
C.2.1. Modificación de la base de datos	88
C.2.2. Modificación del módulo Researcher	89
C.2.3. Modificación del módulo Answer	90
C.3. Encuestas en varios lenguajes	91
C.4. Encuestas con preguntas que saltan a distintas secciones	91
C.4.1. Modificación de la base de datos:	92
C.4.2. Modificación del módulo Researcher:	92
C.4.3. Modificación de la clase StateSurvey	93
C.5. Plataforma en la nube	94
C.5.1. Migración del log	94
C.5.2. Migración a PostgreSQL	95
C.5.3. Servidor web	95
C.6. Consideraciones en el desarrollo de la plataforma	96
D. Análisis de requisitos de la aplicación	97
D.1. Especificación de Requisitos Software	97
D.1.1. Introducción	97
D.1.1.1. Propósito	97
D.1.1.2. Ambito	98
D.1.1.3. Definiciones, Acrónimos y Abreviaturas	98
D.1.2. Descripción general	99
D.1.2.1. Perspectiva del Producto	99
D.1.2.2. Funciones del Producto	99

D.1.2.3.	Características de los usuarios	99
D.1.2.4.	Restricciones	99
D.1.2.5.	Requisitos futuros	100
D.1.3.	Requerimientos específicos	100
D.1.3.1.	Interfaces Externas	100
D.1.3.2.	Requerimientos funcionales	100
D.1.3.3.	Requisitos de Rendimiento	103
D.1.3.4.	Atributos del Sistema	103
E.	Diseño de la aplicación	105
E.1.	Diagrama de Clases	105
E.1.1.	Entidades del sistema	110
E.1.1.1.	User, usuario	110
E.1.1.2.	Survey, encuesta	110
E.1.1.3.	Consent, consentimiento	110
E.1.1.4.	Section, sección	110
E.1.1.5.	Question, pregunta	110
E.1.1.6.	Condition, condición	111
E.1.1.7.	Answer, respuesta.	111
E.1.1.8.	StateSurvey, estado de la encuesta	111
E.1.2.	Entidades del sistema del proyecto "¿Cómo son nuestros volun- tarios?"	111
E.1.2.1.	Raffle, rifa	111
E.1.2.2.	GameImpatience, juego de la impaciencia	112
E.1.2.3.	Game	112
E.2.	Esquema Relacional	112
E.3.	Casos de Uso	115
E.4.	Módulos de la plataforma	117
E.4.1.	Módulo de decoradores	118
E.4.2.	Módulo Planificador de tareas.	119
E.4.3.	Módulo de Autenticación	120
E.4.4.	Módulo del investigador	120
E.4.5.	Módulo del encuestado	125
E.4.6.	Módulo de Feedback	126
E.4.7.	Módulo de estadísticas:	127
E.4.8.	Módulo Juegos:	127

ÍNDICE GENERAL

E.4.9. Módulo de visualización de estadísticas y juegos.	127
E.4.10. Módulo de modelo	128
E.4.11. Módulo de configuración	128
E.4.12. Módulo shell:	129
E.5. Prototipado de las ventanas y cuestiones de usabilidad	130
E.5.1. Diseño del experimento "¿Cómo son nuestros voluntarios?" . .	130
F. ¿Cómo son nuestros voluntarios?	139
F.1. Proyecto "¿Cómo son nuestros voluntarios?"	139
F.2. Introducción y motivación	139
F.3. Ejecución del proyecto	140
F.4. Encuesta	140
F.4.1. Instrucciones	140
F.4.2. Primera parte	141
F.4.2.1. Bloque 1	141
F.4.2.2. Bloque 2	143
F.4.2.3. Bloque 3	143
F.4.2.4. Bloque 4	144
F.4.2.5. Bloque 5	144
F.4.3. Segunda parte	144
F.4.3.1. Bloque 1	145
F.4.3.2. Bloque 2	145
F.4.4. Tercera parte	146
F.4.4.1. Decisión 1	146
F.4.4.2. Decisión 2	149
F.4.4.3. Decisión 3	149
F.4.4.4. Decisiones 4 y 5	150
F.4.4.5. Decisión 6	151
Referencias	152

Índice de figuras

2.1. Tabla comparativa entre las herramientas	8
2.2. Tabla comparativa entre las herramientas	8
2.3. Tabla comparativa entre las herramientas	9
2.4. Tabla comparativa entre las herramientas	9
3.1. Diagrama sencillo que muestra la relación entre el modelo, la vista y el controlador.	12
3.2. web adaptativa	14
6.1. Autenticación mediante correo y contraseña	27
6.2. Identificación mediante OpenID	28
6.3. Control de acceso	29
6.4. Ejemplo de control de acceso	29
6.5. Decorador researcher_required	30
6.6. Página 403	30
7.1. Funciones de modulo Researcher	31
7.2. Ejemplo del controlador index	32
7.3. ejemplo de formulario	32
7.4. Generación dinámica de formularios	33
7.5. Campo likert	34
7.6. Visualización de la escala likert	34
7.7. Validador personalizado	35
7.8. Controlador para editar encuestas	36
8.1. Controlador researcher.index	38
8.2. Plantilla de listado de encuestas	38
8.3. Plantilla nueva encuesta	39
8.4. Vista generada de una nueva encuesta	39
8.5. Vista generada de una nueva encuesta en la que hay un error	40

ÍNDICE DE FIGURAS

8.6. Lenguajes disponibles	40
8.7. Selección de idioma	40
8.8. Captura de la herramienta Lokalize	41
9.1. Resultado de la ejecución de los test unitarios	44
10.1. Esquema del prototipado evolutivo	46
10.2. diagrama de Gantt	47
A.1. Parte del fichero config.py	54
A.2. Ejemplos de comandos disponibles en la shell	57
B.1. Página principal de la aplicación	59
B.2. Página de inicio de sesión mediante OpenID	60
B.3. Página de registro mediante correo	61
B.4. Página de inicio mediante correo/contraseña	61
B.5. Barra de navegación	62
B.6. Página de inicio del módulo de creación de encuestas	63
B.7. Página con los distintos campos de una encuesta	64
B.8. Error mostrado al crear una encuesta	65
B.9. Página de inicio de una encuesta ya creada	66
B.10. Página para crear un nuevo consentimiento	67
B.11. Página con un consentimiento ya creado	68
B.12. Página de nueva sección	69
B.13. Sección ya creada	71
B.14. Creación de preguntas de tipo Si y No	72
B.15. Creación de preguntas de tipo texto	74
B.16. Creación de preguntas de tipo selección	75
B.17. Creación de preguntas de tipo escala likert	77
B.18. Creación de preguntas dependientes de otras preguntas	78
B.19. Ejemplo de pregunta dependiente	79
B.20. Ejemplo de pregunta dependiente	80
B.21. Preguntas para los distintos juegos	81
B.22. Eliminar preguntas	83
E.1. Diagrama de clases en alto nivel de la aplicación	106
E.2. Diagrama de clases de Question	107
E.3. Diagrama de clases de Games	108

E.4. Diagrama detallado de la clase de Game	109
E.5. Esquema relacional	113
E.6. Esquema relacional de los juegos	114
E.7. Diagrama de casos de uso de un voluntario	115
E.8. Diagrama de casos de uso de un investigador	115
E.9. Diagrama de casos del administrador	116
E.10. Ejemplo de traza de eventos genérico	117
E.11. Módulos del sistema	118
E.12. Protección de una vista	119
E.13. Prototipo de creación de una nueva pregunta	131
E.14. Prototipo móvil de creación de una nueva pregunta	132
E.15. Ventana de creación de una nueva pregunta	133
E.16. Despliegue de opciones de creación de una nueva pregunta	134
E.17. Visualización móvil	135
E.18. Menú desplegado	136
E.19. Visualización experimento "¿Cómo son nuestros voluntarios?"	136
E.20. Visualización experimento "¿Cómo son nuestros voluntarios?"	137

ÍNDICE DE FIGURAS

Capítulo 1

Introducción

1.1. Contexto

El proyecto se ha llevado a cabo en el Instituto de Biocomputación y Física de Sistemas Complejos (BIFI), dentro del equipo de la Fundación Ibercivis y en colaboración con los Investigadores Antonio Espín, Universidad de Granada, Pablo Brañas-Garza (Middlesex University) y Anxo Sánchez (Universidad Carlos III)

Ibercivis es una iniciativa internacional de ciencia ciudadana, compuesta por una plataforma de computación voluntaria y por una serie de experimentos que permiten a la sociedad participar en la investigación científica de forma directa y en tiempo real.

Dentro de la computación voluntaria, Ibercivis es una plataforma de computación distribuida, basada en BOINC, que permite a usuarios de internet a participar en proyectos científicos donando ciclos de computación que se emplean para realizar simulaciones y otras tareas.

Ibercivis acerca a la ciudadanía investigaciones punteras y la hace partícipe de la generación de conocimiento científico, al tiempo que dota a la comunidad científica de una potente herramienta de cálculo. El ordenador se convierte en una ventana abierta a la ciencia, creando un canal para el diálogo directo entre investigadores y sociedad.

1.2. Motivación

blablabla..

1. INTRODUCCIÓN

1.3. Objetivos

El objetivo del presente proyecto ha sido el desarrollo de una plataforma para la realización de experimentos sociológico, así como las herramientas necesarias para que los investigadores lleven a cabo el análisis de los datos obtenidos.

Además se ha adaptado esta plataforma para la realización de un experimento cuyo objeto de estudio son los voluntarios de Ibercivis que toman parte en los proyectos de ciencia ciudadana.

La plataforma creada permite a los investigadores la creación de encuestas, pudiendo estas tener distintos niveles de secciones, además de permitir la aleatorización del orden de éstas, así como de las preguntas. Asimismo se puede ponderar el porcentaje de encuestados que realizan una sección concreta.

Por otra parte también se puede definir las fechas en las que una encuesta se encontrará disponible, así como el número máximo de participantes, si es que lo hay. También se puede añadir un tiempo máximo para la realización de la encuesta.

Todas las respuestas son almacenadas en una base de datos junto su tiempo de respuesta, con el objetivo de analizar los tiempos empleados en cada decisión. Toda esta información se puede consultar exportando los resultados a un fichero CSV, del ingles comma-separated values.

La plataforma ha sido creada con el objetivo de ser reutilizable y fácilmente modificable y ampliable para la elaboración de otros experimentos de la misma índole. Tanto los investigadores como los encuestados acceden a la plataforma vía un portal web, teniendo este un diseño web adaptable. Dado el carácter participativo de los experimentos, se ha diseñado para que pueda soportar una carga considerable de usuarios.

Durante el desarrollo de la plataforma se ha tenido en cuenta las necesidades de los investigadores Antonio Espín, Pablo Brañas-Garza y Anxo Sánchez, adaptando tanto la plataforma como el cuestionario *como son nuestros participantes* *2, sobre todo debido a las significativas diferencias de la realización de una encuesta manual a otra de manera telemática.

La encuesta consta de tres partes diferenciadas. Una primera parte que consiste en una serie de preguntas sobre distintos aspectos de la personalidad. Las otras dos partes representan diversos escenarios en los que se tienen que tomar una serie de decisiones: en unos casos las decisiones solo dependen del propio usuario, mientras que en otros también dependerá de la elección de otro participante anónimo escogido al azar entre el resto de participantes. Estos escenarios están basados en distintos

juegos experimentales de economía, como puede ser el juego del dictador o el juego del ultimatum. Tanto la segunda como la tercera parte, las cuestiones se plantean en términos de una ganancia económica. En casos elegidos de forma aleatoria, el voluntario recibe un pago asociado a una de sus decisiones.

El objetivo final del experimento es permitir a los investigadores comparar a los voluntarios de Ibercivis con la muestra de población general del trabajo "Experimental subjects are not different" [F. Exadaktylos, A. M. Espín & P. Brañas-Garza, Sci. Rep. 3, 1213 (2013)] y caracterizar a una población amplia de voluntarios para después solicitar su colaboración en futuros proyectos seleccionándolos conforme a su perfil.

1.4. Estructura de la memoria

En la presente memoria se describe el proceso llevado a cabo para el desarrollo de la plataforma. Su contenido se ha distribuido en *XX* capítulos, siendo estos:

- Capítulo 1, este capítulo de introducción.
- Capítulo 2 se muestra la elección de la tecnología implicada en la realización de la plataforma.
- Capítulo 3, análisis de las herramientas existentes relacionadas con el proyecto
- Capítulo n, las conclusiones

1. INTRODUCCIÓN

Capítulo 2

Estado del Arte

Antes de la realización de la aplicación se estudió los requerimientos que debía cumplir la plataforma, además se estudió las distintas alternativas libres que existen para la elaboración de cuestionarios y si estas se adaptaban a los requerimientos.

2.1. Análisis de requerimientos

Para ser capaces de evaluar las distintas alternativas existentes, antes debemos realizar en primer lugar un análisis de los requerimientos necesarios para la realización de nuestra plataforma. Estos requerimientos han sido fruto del estudio del documento ****"¿Cómo son nuestros voluntarios?"****, así como las distintas conversaciones con los investigadores. Cabe destacar que estos requerimientos se han ido ampliando durante el desarrollo de la plataforma, a medida que se ha ido mostrando el aspecto de la plataforma y el cuestionario en versión telemática ****"¿Cómo son nuestros voluntarios?"****, ya que este mismo cuestionario se ha ido adaptando a las necesidades y a las diferencias que existen entre la elaboración de un cuestionario de manera escrita, a otra de manera telemática.

Los requerimientos mínimos que debía cumplir el cuestionario fueron los siguientes:

- Poder evaluar y calcular el tiempo empleado de cada usuario al contestar cada pregunta.
- El orden de las distintas secciones debía seguir una pauta muy específica, existiendo secciones con un orden fijo y las otras secciones siguiendo una pauta semialeatoria, ya que hay secciones que deben ir contiguas.
- Hay secciones excluyentes entre si.
- Los usuarios tienen un tiempo limite para la realización del cuestionario.

2. ESTADO DEL ARTE

- El cuestionario solo debe estar presente para su realización en un rango de tiempo predefinido.
- Existe un numero máximo de participantes.
- Preguntas que deben ser contestada según la respuesta dada a otra anterior.
- Preguntas de validación, en la que se comprueba que el usuario ha contestado bien y se le notifica si la respuesta es correcta o no, con un número máximo de intentos.
- Disponibilidad de una interfaz gráfica.

En el anexoX se encuentra la lista final de los requisitos de la aplicación.

2.2. Herramientas actuales

La importancia de la web ha ido creciendo de manera exponencial conforme han ido pasando los años. Por lo que se han ido creando multitud de herramientas para ir resolviendo las distintas necesidades que se han ido creando y repitiendo a lo largo de los años. En el caso de las encuestas no es distinto al de los otros, pero debido a nuestras necesidades concretas que hemos expuesto anteriormente nos hemos centrado sobretodo en las alternativas libres o que expusiesen un API para su integración con la plataforma.

2.2.1. Surveyor

Surveyor es una herramienta para crear encuestas, cuestionarios y premios e integrarlos en una aplicación escrita en Ruby on Rails. Usa una licencia MIT, compatible con GPL. En un principio fue diseñado para ofrecer estudios de investigación clínicos en grandes poblaciones.

Para la elaboración de encuestas hace uso de un Lenguaje específico del dominio, permitiendo una escritura fácil y rápida de encuestas. Además permite la personalización del modelo, vista y controlador, así como los routes.

Como parte negativa, no tiene un soporte consistente para HTML, así como un código poco documentado y ninguna guía de desarrollo.

2.2.2. LimeSurvey

Es una herramienta escrita en PHP bajo la licencia GNU GPL. Entre sus usuarios se encuentra proyectos como OpenOffice, ubuntu o GNOME.

En si es una plataforma muy completa, con un potente editor WYSIWYG What You See Is What You Get, "lo que ves es lo que obtienes", así como numerosas opciones en la elaboración de encuestas, como puede ser el restringir el numero máximo de participantes según sus característica(rango de edad, poblacion, sexo..)

Su gran inconveniente es que está pensado para usarla en si misma, ya que trae todo integrado, haciendo difícil la modificación fuera de las opciones predeterminadas. Es mas su modelo de negocio es el ofrecer hosting de LimeSurvey a cambio de una tarifa que depende del número de encuestados.

2.2.3. Survey Project

Es una herramienta bajo GPL2 escrita en C# y soporte único para máquinas con un sistema operativo windows.

Tanto a nivel de características como de modelo de negocio, es muy parecido a LimeSurvey.

2.2.4. Google Forms

Google posee una herramienta para la elaboración de formularios, así como una API para poder acceder ella, tanto para la elaboración de formularios como para obtener los resultados de ella. Además de esta herramienta, también posee un servicio para la elaboración de encuestas, Google Surveys, con un precio entre 0,10\$ y 3,5\$ por usuario que complete el cuestionario, dependiendo de la longitud de este.

En la elaboración de formularios, posiblemente la solución de Google sea la mas accesible de todas, sobre todo por su cuidado en el aspecto y las funcionalidades a la hora de crear formularios. A parte de tener todo integrado, exportándose automáticamente los resultados a GoogleDocs y pudiendo obtener un análisis de estadísticas sencillo. Por otra parte es la mas limitada a la hora de crear formularios complejos y sin la posibilidad de modificación.

2.2.5. Comparativa

Del estudio de las herramientas actuales se obtuvo bastante información interesante, sobre todo las características deseables para la elaboración de una plataforma para

2. ESTADO DEL ARTE

Tipo de preguntas /Plataforma	Surveyor	LimeSurvey	Survey Project	Google Forms	Swarm Surey
Numérica	Si	Si	Si	Si	Si
Decimal	Si	Si	Si	Si	Si
Texto	Si	Si	Si	Si	Si
Expresión regular	No	Si	Si	Si	Si
Escala likert	No	Si	Si	Si	Si
Radio boton	Si	Si	Si	Si	Si
Choice	No	Si	Si	Si	Si
Preguntas obligatorias	No*	Si	Si	Si	Si
Preguntas dependientes	Si	Si	Si	No	Si
Preguntas de validación	No	No	No	No	Si

Figura 2.1: Tabla comparativa entre las herramientas

Lógica del programa /Plataforma	Surveyor	LimeSurvey	Survey Project	Google Forms	Swarm Surey
Secciones aleatorias	No	No	No	No	Si
Secciones excluyentes	No	No	No	No	Si
Cálculo de tiempos	No	No	No	No	Si
Tiempo limite	No	No	Si	No	Si
Fecha limite	No	Si	No	No	No
Número máximo de participantes	No	Si	Si	No	Si

Figura 2.2: Tabla comparativa entre las herramientas

la elaboración de encuestas, esta información recogida así como los requisitos necesarios para nuestra plataforma fueron los usados para definir los requerimientos finales. Observando las tablas de la figura 2.1 se puede comparar el grado de cumplimiento de nuestra plataforma respecto a las otras.

Mirando únicamente la tabla, Surveyor puede parecer que sea de las peores elecciones, sobre todo si solo se comparan funcionalidades, pero analizando un poco más la plataforma sería la mejor opción para adaptarla, debido a su tamaño y a que ofrece un núcleo básico para usarse de base, pero debido a su nula documentación, se estimo que el tiempo necesario para comprender en profundidad el funcionamiento de esta, así como para adaptar todas las características necesarias para nuestra plataforma y elaboración del experimento iba a ser mayor que el necesario para crear una desde cero. Para no caer en el error de Surveyor se ha intentado documentar lo mejor posible la plataforma, además de la elaboración de una guía rápida de desarrollo en el que se incluyen ejemplos de como ampliar la plataforma con una serie de características que no posee.²

2.2 Herramientas actuales

Ampliación de la Plataforma/ Plataforma	Surveyor	LimeSurvey	Survey Proyect	Google Forms	Swarm Surey
Facilidad de ampliación	Media	Baja	Baja	No	Alta
Facilidad de desarrollo	Media	Baja	Baja	Media, usando la API	Alta
Sistema de migración de bbdd	No	No	No	No	Si
Shell	No	No	No	No	Si
Manual de desarrollo	No	No	No	Si	Si

Figura 2.3: Tabla comparativa entre las herramientas

Otras cuestiones /Plataforma	Surveyor	LimeSurvey	Survey Proyect	Google Forms	Swarm Surey
Interfaz gráfica	No	Si	Si	Si	Si
Lenguaje específico del dominio	Si	No	No	No	No
Exportar importar a fichero	Si	Si	Si	No	Si
Multilenguaje	Si	Si	Si	Si	No ¹
Editor de texto amigable	No, uso inconsistente de html	No	No	No	Si, uso de syntaxis Markdown
Plantillas	Si	Si	Si	No	Si
Uso de direcciones web amigables	Si	No	No	No	Si
Multiplataforma	Si	Si	No	No	Si
Uso de nube	No	No	No	No	No

Figura 2.4: Tabla comparativa entre las herramientas

2. ESTADO DEL ARTE

Capítulo 3

Contexto Tecnológico

En este capítulo se estudia las diferentes tecnologías que fueron utilizadas en el desarrollo del presente PFC. En primer lugar se explican algunos conceptos referente al modelo-vista-controlador (MVC). Después se introduce brevemente el lenguaje utilizado así como las tecnologías asociadas. Por último el software utilizado durante las fases de desarrollo del proyecto.

3.1. Modelo-vista-controlador

MVC es un patrón de diseño de las aplicaciones software, según la definición de Christopher Alexander ^{**}"cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces."^{**} ^{*1}*1 "Design Patterns" [GoF95] en la sección "¿Qué es un patrón de diseño?"

El patrón MVC consiste en tres tipos de objetos:

- El **Modelo** es la representación de la información con la cual es sistema opera.
- La **Vista** que es su representación del modelo (información y lógica del negocio), en un formato adecuado para interaccionar con él.
- El **Controlador** que define el modo en que la interfaz reacciona a la entrada del usuario e invoca peticiones al modelo.

La figura 3.1 muestra la relación entre el modelo, la vista y el controlador. Las líneas solidas indicas una asociación directa, las punteadas indirectas.

En otras palabras, el flujo de control sería:

- ^{*}El usuario realiza una acción en la interfaz.

3. CONTEXTO TECNOLÓGICO

- El controlador trata el evento de entrada.
- El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio en el estado del modelo.
- Se genera una nueva vista. La vista toma los datos del modelo. En ningún momento, el modelo tiene conocimiento directo de la vista.
- La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.

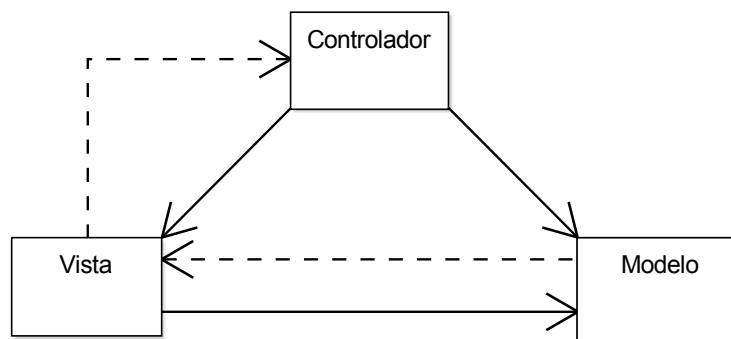


Figura 3.1: Diagrama sencillo que muestra la relación entre el modelo, la vista y el controlador.

3.2. Flask

Dentro de los objetivos del proyecto, se decidió que tanto los usuarios como los investigadores interaccionar con la plataforma mediante un portal web. Para la creación de esta plataforma, se eligió Flask, que es una micro plataforma para el desarrollo web escrito en Python. El significado de *micro* es que tiene como objetivo mantener un núcleo simple pero extensible, sin tomar decisiones por el desarrollador.

Flask no incluye capa de abstracción para la base de datos, validación de formularios o cualquier otra cosa que se pueda encontrar mediante el uso de diferentes bibliotecas que ya existen. En su lugar Flask soporta extensiones para añadir estas funcionalidades a la aplicación.

El núcleo de Flask trae por defecto:

- Un servidor de desarrollo y un debugger incluido en el servidor.
- Soporte de test unitarios integrados.

- Uso de URLs amigables.
- Generador de plantillas Jinja2.
- Soporte para cookies seguras.

La decisión de usar Flask como marco de trabajo para el proyecto viene debido a las características anteriormente citadas, además de su suave curva de aprendizaje, permitiendo de manera rápida empezar a construir el sistema con tener unos conocimientos básicos de Python y HTML.

Por otra parte Flask posee una licencia de software libre permisiva, concretamente BSD, dándonos gran libertad a la hora de desarrollar. También posee una excelente documentación con diversos consejos como puede ser la estructuración de un programa grande, o el manejo de ficheros de configuración.

Todo esto facilita la creación de aplicaciones de manera fácil y rápida.

3.3. Python

Debido a la elección de Flask, para el desarrollo de la plataforma se ha usado el lenguaje de programación Python, desarrollado a finales de los ochenta por Guido van Rossum en el Centro para las Matemáticas y la Informática en los Países Bajos. Python es un lenguaje de programación multiparadigma y que soporta la orientación a objetos, programación imperativa y la programación funcional. La sintaxis de Python esta basada en la legibilidad y transparencia, obligando al usuario a escribir código bien indentado.

3.4. Jinja2

Jinja2 es un lenguaje moderno y amistoso para la creación de plantillas. El uso de plantillas, junto con los datos obtenidos del modelo, permite la generación de paginas web dinámicas de manera fácil, permitiendo con una única plantilla, la generación de páginas diferentes.

El motor de Jinja2 entre sus características están el uso de filtros, bucles variables, herencia, así como el uso de funciones externas escritas en Python. Además provee de un sistema automático de escapado HTML, para prevenir técnicas de XSS**, que es una brecha de seguridad típico de las aplicaciones web.

Jinja2 junto con el código obtenido del modelo, es capaz de generar la vista del modelo MVC, esta vistas son páginas web HTML que recibe el usuario.

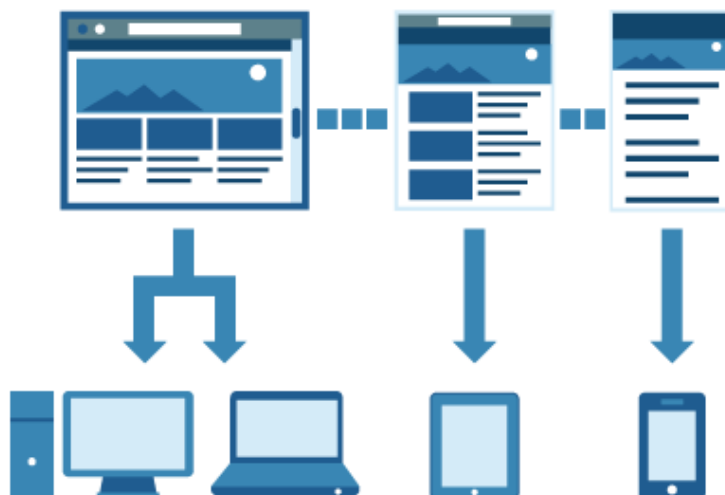


Figura 3.2: web adaptativa

3.5. Otros lenguajes

Aparte de los lenguajes indicados anteriormente, se ha usado HTML, que es el lenguaje estándar que sirve para la elaboración de páginas web.

También se ha hecho uso de CSS, Hojas de Estilo en Cascada (Cascading Style Sheets), que es un lenguaje usado para describir el aspecto y el formato de un documento escrito en un lenguaje de marcas, en nuestro caso HTML.

Además se ha hecho uso de JavaScript, un lenguaje de programación que normalmente se usa en el lado del cliente, en nuestro caso navegador web, que permite mejoras y la adición de funcionalidades en la interfaz del usuario, vista.

3.6. Bootstrap

En los objetivos de la plataforma se incluía que el diseño web fuese adaptable, esto significa que el diseño de la página se ajuste dinámicamente a las características del dispositivo usado, ya sea este un ordenador de escritorio, una tablet, teléfono móvil...

Para ello se hizo uso de Bootstrap, que es una colección de herramientas que contiene código HTML, plantillas CSS, formularios y otros componentes de la interfaz de una página web para la creación de páginas web con diseño adaptable.

3.7. SQLAlchemy

Para dar persistencia a los datos manejados por el modelo, se hizo uso de SQLAlchemy, que es un ORM, Mapeo objeto-relacional (Object Relational Mapper). El ORM es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos, en nuestro caso Python, y la utilización de una base de datos relacional, creando en la practica una base de datos orientada a objetos.

El uso de SQLAlchemy permite trabajar con bases de datos relacionales de diferentes fabricantes, abstrayéndote de la elección usada, como puede ser SQLite, MySQL u Oracle, pudiendo escribir la misma consulta independientemente de la base de datos usada por debajo. Lo cual permite cambiar fácilmente de base de datos a usar.

Además también se ha hecho uso de Alembic, que es una herramienta de migración de base de datos para SQLAlchemy, esta herramienta permite ir actualizando de manera transparente la base de datos mientras el programa va creciendo, sin necesidad de crear una nueva base de datos y realizar costosas migraciones de una a otra. Además de mantener un historial de los cambios en la base de datos, lo cual permite revertir la base de datos a una anterior versión. Todo esto facilita el desarrollo de la aplicación, así como la actualización de esta en entornos de producción.

3.8. WTForms

Para la realización de los cuestionarios se ha utilizado WTForms, el cual es una biblioteca para la validación y representación de formularios, abstrayéndote del código HTML necesario para la representación de estos elementos en la pagina web.

Además permite de manera fácil la creación de nuevos validadores y la creación de nuevos campos hechos a medida para la aplicación.

3.9. GitHub

Ya que una de las motivaciones del proyecto es la realización de una plataforma que puede ayudar a la comunidad científica a la elaboración de experimentos participativos, se determinó que el desarrollo del proyecto fuese completamente transparente, libre y abierto.

Para ello se usó desde el principio GitHub, el cual es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de versiones Git. Todo el código, así como la evolución de este se puede encontrar en "https://github.com/nukru/frame_game_th

3. CONTEXTO TECNOLÓGICO

3.10. Software de apoyo

Durante el desarrollo del proyecto se utilizaron además las siguientes aplicaciones:

- Sublime Text 2 como editor de texto y editor de código fuente, junto las siguientes extensiones:
 - Git: extensión para integrar Git dentro de Sublime Text.
 - Indent XML: indentación de código XML.
 - Sublime Code Intel, una extensión para el autocompletado de código.
 - Sublime Linter, para la comprobación de errores potenciales en la escritura de código.
- JMeter para la realización de pruebas de sobrecarga.
- L^AT_EX para la elaboración de la documentación. Los componentes utilizados fueron:
 - T_EX Live como distribución de T_EX
 - L_YX como entorno gráfico
 - OCIAAM Thesis for L_YX como plantilla.

Capítulo 4

Esqueleto de la aplicación construida

En este apartado vamos a indicar el esqueleto sobre el que hemos construido la aplicación, haciendo uso de las distintas tecnologías mencionadas en el contexto tecnológico.

4.1. Flask

Siguiendo los distintos consejos dados en la documentación de Flask, así como las recomendaciones de patrones para Flask, la estructura de nuestra plataforma de encuestas es la siguiente:

SwarmSurvey - Directorio raíz de la aplicación

manage.py - Se encarga de la gestión de la aplicación, ya sea ejecutar el servidor, migrar la base de datos, o entrar en la interfaz de línea de comandos de la aplicación

config.py - Fichero de configuración base, usando un modelo de herencia de clases para la configuración, así como el uso de variables de entorno para cambiar de configuración

app

auth - Módulo para el registro y autenticación de los usuarios

__init__.py - Fichero de inicialización del módulo

forms.py - Formularios que se mostrarán en las plantillas para el registro y autenticación de usuarios

validators.py - Fichero con los distintos validadores creados para comprobar los formularios

4. ESQUELETO DE LA APLICACIÓN CONSTRUIDA

views.py - Fichero con la resolución de rutas y las funciones expuestas a través de estas rutas (eg: /login, /register...)

feedback - Módulo que da soporte de feedback al experimento "¿Cómo son nuestros voluntarios?"

...

function_jinja - Módulo que contiene distintas funciones creadas para el generador de plantillas jinja2

__init__.py

functions.py

game - Módulo que contiene la lógica a la hora de seleccionar los usuarios para los juegos del experimento "¿Cómo son nuestros voluntarios?"

__init__.py

game.py - Juegos del apartado 3 del experimento "¿Cómo son nuestros voluntarios?"

game_impatience.py - Juegos del apartado 2 del experimento "¿Cómo son nuestros voluntarios?"

raffle.py - Rifa del experimento "¿Cómo son nuestros voluntarios?"

main -Módulo que contiene la información básica de la plataforma

__init__.py

errors.py - Personalización de los distintos errores que puede lanzar la plataforma, ya sean por parte del cliente(4xx) o del servidor(5xx)

views.py - Vistas de la ruta raíz (/), así como el selector de idioma.

researcher - Módulo para la creación de encuestas por parte de los investigadores

...

scheduler - Módulo para la planificación de tareas en el tiempo

...

surveys - Módulo para visualizar y contestar encuestas por parte de los usuarios.

...

static - Directorio que contiene los elementos estáticos de la plataforma web, como puede ser el uso de imágenes, css...

css - Directorio que contiene los ficheros CSS de la aplicación

img - Directorio que contiene las imagenes de la aplicación

js - Directorio que contiene los ficheros JavaScript de la aplicación

text - Directorio que contiene ficheros de texto de la aplicación

stats - Módulo que contiene la generación y visualización de estadísticas y resultados de las encuestas así como del experimento "¿Cómo son nuestros voluntarios?"

...

templates - Directorio con las distintas plantillas usadas para la generación de las distintas páginas web

auth - Directorio con las plantillas del módulo auth

login.html - Plantilla de inicio de sesión a través de OpenID

register.html - Plantilla de registro

loginEmail.html - Plantilla de inicio a través de correo/contraseña

feedback - Directorio con las plantillas de modulo de feedback

...

...

translations - Directorio con los distintos idiomas que soporta la plataforma

es - Traducción al español

en - Traducción al ingles

__init__.py - Fichero de inicialización de la plataforma web

decorators.py - Fichero con las funciones para la protección de las distintas vistas de la plataforma

models.py - Declaración del modelo de clases que hace uso de la base de datos.

...

migrations - Directorio que contiene la información necesaria para la migración de la base de datos, este directorio se genera automáticamente.

...

tests - Directorio que contiene los distintos tests unitarios del sistema

4. ESQUELETO DE LA APLICACIÓN CONSTRUIDA

`test_models.py` - Fichero con los test del modelo

...

`logs` - Directorio con los logs de la aplicación

4.2. Modelo Vista Controlador implantado

A continuación se va a explicar brevemente como se trabaja con el MVC implantado en la plataforma.

El motor de Flask recoge todas las peticiones http que llegan al servidor y busca si existe alguna función para la ruta recibida (ficheros `views.py`). Si el patrón encaja con una ruta reconocida, ejecuta la función del "Controlador" correspondiente a esa ruta. El controlador es una función en Python que al final termina devolviendo una respuesta HTTP a la petición recibida. La respuesta suele ser una página web, la cual se genera a través de una plantilla (ficheros del directorio `templates`), normalmente para la generación de esta "Vista", se necesita información de la base de datos (fichero `models.py`), esto sería el "Modelo"

4.3. Plataforma Modular

Como se ha visto en el esqueleto de la aplicación, existen varios módulos en la aplicación, cada uno con sus ficheros "*views.py*", que contienen las rutas y los controladores. Para facilitar esta modulación se utiliza el concepto de *blueprints* en Flask, simplificando en gran medida la creación de grandes aplicaciones, al poder extender la aplicación a través de módulos.

Aparte de lo anteriormente citado, el uso de *blueprints* nos permite:

- Registrar un módulo en un subdominio o prefijo URL, incluyendo de manera transparente las rutas del fichero `views`, al subdominio o prefijo URL.
- También permite registrar varias veces el mismo módulo en diferentes subdominios o prefijos URL.
- Los *blueprints* se registran al inicio de la aplicación, por lo que se puede implementar distintas funcionalidades como extensiones de la plataforma principal

Por ejemplo, para registrar el módulo `auth` bajo la ruta `/auth` y `/autenticación`, simplemente hace falta añadir al fichero de inicialización de la plataforma las siguientes líneas:

```
1 from .auth import auth as auth_blueprint
2 app.register_blueprint(auth_blueprint, url_prefix='/auth')
3 app.register_blueprint(auth_blueprint, url_prefix='/autenticacion')
```

<http://flask.pocoo.org/docs/blueprints/#blueprints>

<http://flask.pocoo.org/docs/> <http://flask.pocoo.org/docs/patterns/>

<http://flask.pocoo.org/docs/config/#development-production> <http://flask.pocoo.org/docs/con>
from-files

4. ESQUELETO DE LA APLICACIÓN CONSTRUIDA

Capítulo 5

Enrutamiento

El diseño de rutas de la plataforma, nos permite tener una visión clara de la funcionalidad de la aplicación.

Como ya hemos explicado anteriormente en el esqueleto de la aplicación, la plataforma está subdividida en varios módulos, a cada módulo se accede mediante una ruta distinta. A la hora de elegir las rutas siempre se ha tenido en cuenta el requisito de que sean amigables y fácilmente recordables.

A continuación veremos como Flask resuelve las rutas, así como un resumen del mapa de rutas de la aplicación.

5.1. Resolución de rutas en Flask

Las rutas son las URL o localizador de recursos uniforme que nos permite acceder a las distintas funcionalidades de la plataforma.

Flask posee tres formas de definir las reglas de las rutas en el sistema:

- Usando el decorador *flask.Flask.route()*
- Usando la función *flask.Flask.add_url_rule()*
- Accediendo directamente al sistema de rutas de Werkzeug, el cual está expuesto mediante *flask.Flask.url_map*

Para el desarrollo de la plataforma se ha elegido la opción del decorador, debido a la sencillez de uso y a la abstracción de la función y la ruta usada para esta.

La ruta se define mediante una cadena que simboliza la regla de una URL, la cual puede contener variables, de tal manera que la variable de la URL se le pasará a la función de Python que está decorando. Además en estas rutas se puede limitar el método de petición HTTP (GET, POST, PUT...).

5. ENRUTAMIENTO

En el siguiente ejemplo vemos que para acceder a la vista que nos permite modificar una encuesta, debemos usar la ruta `/survey/N`, siendo `N` el número de la encuesta, además podemos acceder mediante los métodos `GET` y `POST`. También se puede acceder mediante la cadena `"titulo_de_la_encuesta_id_survey"`, para facilitar el acceso.

```
1 @blueprint.route('/survey/<int:id_survey>', methods = ['GET', 'POST'])
2 @blueprint.route('/survey/<string:title_survey>', methods = ['GET', 'POST'])
3 ...
4 def editSurvey(id_survey):
```

<http://flask.pocoo.org/docs/api/#url-route-registrations>

5.2. Rutas de la aplicación

Como se explica en la sección 4.3, la plataforma creada tiene distintos módulos, cada uno registrada en un prefijo URL distinto, gracias al uso de *blueprints*, a continuación vamos a detallar las distintas rutas de la aplicación dividida en módulos.

5.2.1. Módulo main

Todas las direcciones de este módulo están bajo la raíz de la dirección web y están disponibles para todos los usuarios.

`/` - Raíz de la aplicación
`/main` - Raíz de la aplicación

5.2.2. Módulo de autenticación

Todas las direcciones de este módulo están bajo la ruta `/auth` y están disponibles para todos los usuarios.

`/` - Autenticación mediante OpenId
`/login` - Autenticación mediante OpenId
`/loginEmail` - Autenticación mediante correo/contraseña
`/register` - Registro de cuenta de usuario
`/logout` - Cierre de la sesión actual

5.2.3. Módulo de investigador

Todas las direcciones de este módulo están bajo la ruta */researcher* y solo pueden acceder aquellos usuarios que sean investigadores.

- / - Listado de las encuestas de un investigador
- /index - Listado de las encuestas de un investigador
- /new - Creación de una nueva encuesta
- /survey/<survey> - Modificación de una encuesta
- /survey/deleteSurvey/<survey> - Eliminación de una encuesta
- /survey/exportSurvey/<survey> - Exportación de una encuesta
- /survey/<survey>/consent/add - Creación de un consentimiento
- /survey/<survey>/consent/<consent>/delete - Eliminación un consentimiento
- /survey/<survey>/consent/<consent> - Modificación de un consentimiento
- /survey/<survey>/section/new - Creación de una sección nueva
- /survey/<survey>/section/<section> - Modificación de una sección
- /survey/<survey>/deleteSection/<section> - Eliminación de una sección
- /survey/<survey>/duplicateSection/<section>/section/<section2> - Duplicación de una sección
- /survey/<survey>/duplicateSection/<section>/survey/ - Duplicar una sección
- /survey/<survey>/section/<section>/new - Creación de una subsección
- /survey/<survey>/section/<section>/addQuestion - Creación de una pregunta
- /survey/<survey>/section/<section>/question/<question> - Modificación de una pregunta
- /survey/<survey>/Section/<section>/deleteQuestion/<question> - Eliminación de una pregunta
- /survey/exportStats/<survey> - Exportación de las estadísticas de una encuesta

5.2.4. Módulo del encuestado

Todas las direcciones de este módulo están bajo la ruta */survey* y está disponible para todos los usuarios que hayan iniciado sesión.

- / - Listado de las encuestas

5. ENRUTAMIENTO

`/index` - Listado de las encuestas

`/<survey>` - Inicio de una encuesta

`/<survey>/consent` - Muestra el consentimiento de una encuesta

`/<survey>/consent/<consent>` - Muestra el consentimiento de una encuesta

`/<survey>/section/<section>` - Muestra una sección de una encuesta para que el encuestado la responda

5.2.5. Módulo de feedback

Todas las direcciones de este módulo se encuentran bajo la ruta `/feedback` y está disponible para todos los usuarios que hayan iniciado sesión.

`/<survey>` - Muestra feedback sobre las decisiones tomadas en la encuesta

`/<survey>/<feedback>` - Muestra feedback sobre las decisiones tomadas en la encuesta

5.2.6. Módulo de visualización de estadísticas

Todas las direcciones de visualización de estadísticas se encuentran bajo la ruta `/stats` y está disponible para todos los usuarios que sean investigadores.

`/` - Listado de las encuestas disponibles

`/index` - Listado de las encuestas disponibles

`/<survey>` - Listado de los resultados de una encuesta

`/<survey>/games` - Listado de los juegos disponibles de una encuesta

`/<survey>/games/<game>` - Listados de los resultados de los juegos de una encuesta

Chapter 6

Seguridad

En el capítulo 5 pudimos ver el listado de todas las rutas de acceso disponibles. Algunas de estas rutas son algo especiales, ya que solo están disponibles para una serie de usuarios con ciertos permisos. En este capítulo vamos a tratar como se gestiona la autenticación y los permisos de acceso a las distintas rutas web del programa.

6.1. Autenticación de usuarios

Para la autenticación de usuarios se ha hecho uso de la extensión Flask-Login, el cual proporciona una gestión básica de sesiones de usuarios para Flask, manejando tareas tan comunes como inicio, cierre e identificación automática de sesiones.

Existen dos formas de autenticación:

- Mediante correo y contraseña, como se puede ver en la figura 6.1, se lee de un formulario, el correo y la contraseña introducida por el usuario, primero se verifica que el usuario existe, y luego se comprueba que el hash de la contraseña es correcta.

```
1 @blueprint.route('/loginEmail', methods=['GET', 'POST'])
2 def loginEmail():
3     form = LoginFormEmail()
4     if form.validate_on_submit():
5         user = User.query.filter_by(email=form.email.data).first()
6         if user is not None and user.verify_password(form.password.data):
7             login_user(user, form.remember_me.data)
8             return redirect(request.args.get('next') or url_for('main.index'))
9         flash(gettext('Invalid email or password.'))
10    return render_template('auth/loginEmail.html', form=form)
```

Figure 6.1: Autenticación mediante correo y contraseña

6. SEGURIDAD

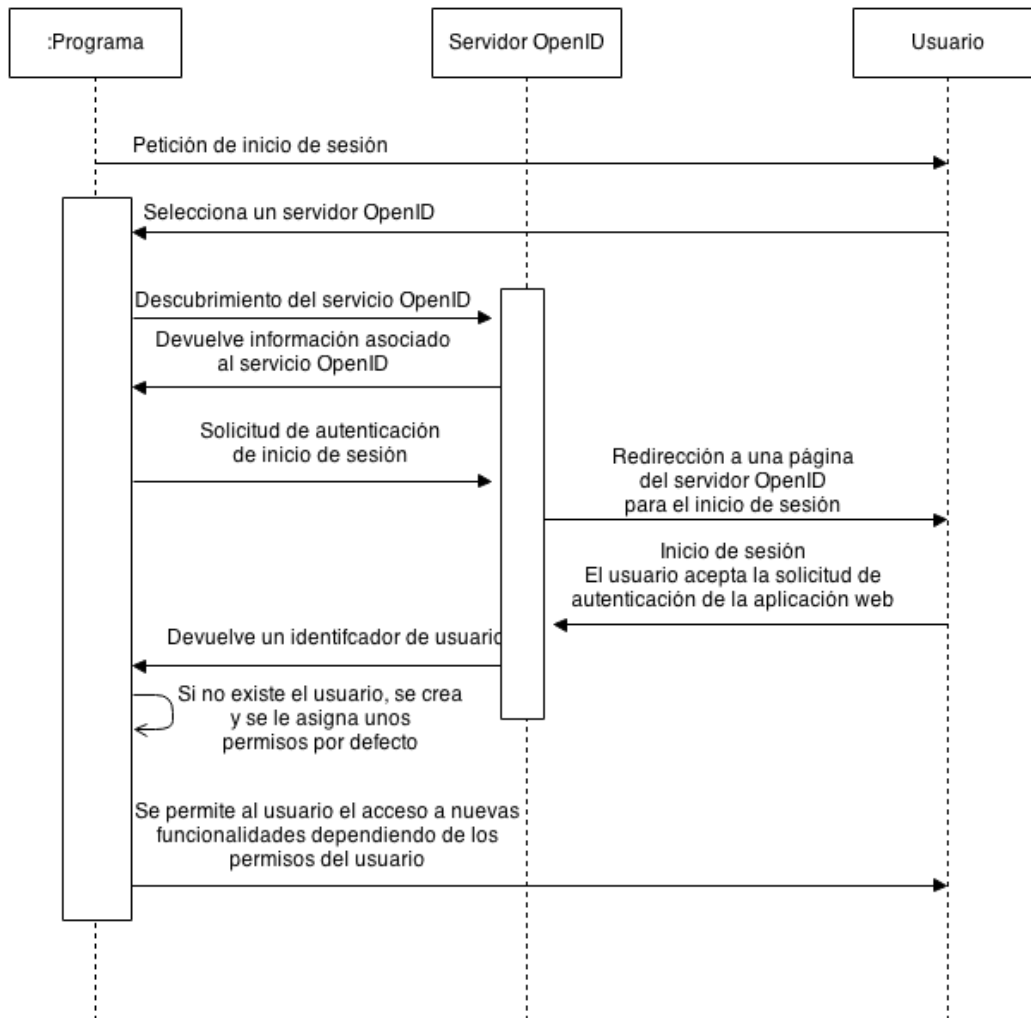


Figure 6.2: Identificación mediante OpenID

- Mediante OpenID, el programa se intenta autenticar con el servidor OpenID solicitado por el usuario, como se puede ver en la figura 6.2

6.2. Control de acceso

Debido a que existen ciertas funcionalidades a las que solo pueden acceder ciertos usuarios, hay que implementar un control de acceso a las distintas rutas web, las cuales se han detallado en el capítulo 5.

Para ello, antes de cada petición se comprueba que el usuario se ha autenticado y de que usuario se trata, como se puede ver en la figura 6.3.

Luego, las vistas de las distintas rutas se han protegido mediante decoradores, que comprueban entre otras cosas los permisos del usuario. Los decoradores disponibles

```
1 @blueprint.before_app_request
2 def before_request():
3     if current_user.is_authenticated():
4         g.user = current_user
5     else:
6         g.user = None
```

Figure 6.3: Control de acceso

```
1 [code]
2 @blueprint.route('/survey/new', methods = ['GET', 'POST'])
3 @login_required
4 @researcher_required
5 def new_survey():
6 [/code]
```

Figure 6.4: Ejemplo de control de acceso

son los siguientes:

- `login_required`: Comprueba que el usuario ha ingresado en la plataforma.
- `researcher_required`: Comprueba que el usuario es un investigador.
- `finished_survey`: Comprueba que el usuario ha finalizado la encuesta.
- `belong_researcher`: Valida que la encuesta, sección o pregunta que quiere modificar pertenece al investigador en cuestión y no a otro.
- `valid_survey`: Comprueba que la encuesta ya se ha publicado.
- `there_is_stateSurvey`: Valida que existe y es valido el estado de la encuesta por parte de un usuario que quiere rellenar una encuesta.

Por ejemplo, en la figura 6.4, se puede ver como se protege la vista de creación de una nueva encuesta, a la cual se puede acceder mediante la dirección `http://servidor/survey/new`, antes de poder acceder a la vista, se comprueba que el usuario se haya autenticado en el sistema, mediante el decorador `@login_required`, y que además el usuario tenga permisos de investigador, `@researcher_required`. De esta forma se puede proteger las páginas a los que los usuarios pueden acceder.

Si el usuario no tiene los permisos necesarios, figura 6.5, se le redirige a página HTTP personalizada con el código de estado 403, figura 6.6

6. SEGURIDAD

```
1 def researcher_required(f)
2     """Checks if the user is and researcher or not"""
3     @wraps(f)
4     def decorated_function(*args, **kwargs):
5         if current_user.is_researcher():
6             return f(*args, **kwargs)
7         else:
8             return abort(403)
9     return decorated_function
```

Figure 6.5: Decorador researcher_required

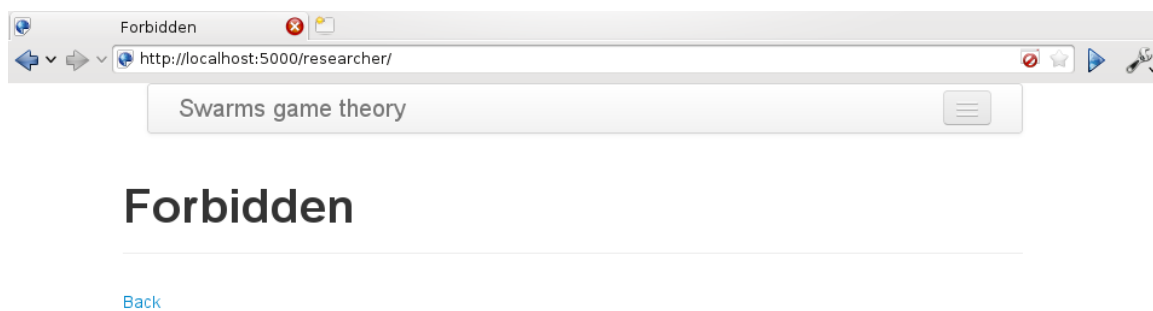


Figure 6.6: Página 403

Chapter 7

Controlador

Durante la memoria se ha ido utilizando el termino modelo, vista y controlador, siendo el controlador la función que ejecuta la aplicación cuando recibe una petición con un patrón reconocido en las rutas de la aplicación web. Este capítulo tratará de presentar la estructura básica de un controlador de la aplicación.

7.1. Estructura.

Como se puede observar en el capítulo 4, cada módulo contiene un fichero *views.py*, que reúne las funciones públicas que se ocupa de responder ante las rutas web.

Por ejemplo, en el módulo de creación de encuestas, ver figura 7.1, podemos ver las distintas funciones públicas a las que podemos acceder mediante una ruta web, cada función iría precedida de la ruta por la que se accede a dicha función, así como los permisos necesarios para acceder, todo esto explicado en los capítulos 5 y 6

```
1 def index():
2 def new():
3 def edit_survey(id_survey):
4 def delete_survey(id_survey):
5 def export_survey(id_survey):
6 def add_consent(id_survey):
7 def delete_consent(id_survey, id_consent):
8 def edit_consents(id_survey, id_consent):
9 def edit_section(id_survey, id_section):
10 def delete_section(id_survey, id_section):
11 def duplicate_section(id_survey, id_section):
12 def add_subsection(id_survey, id_section):
13 def add_question(id_survey, id_section):
14 def edit_question(id_survey, id_section, id_question):
15 def delete_question(id_survey, id_section, id_question):
16 def export_stats(id_survey):
```

Figure 7.1: Funciones de modulo Researcher

7. CONTROLADOR

```
1 @blueprint.route('/')
2 @blueprint.route('/index')
3 @login_required
4 @researcher_required
5 def index():
6     surveys = Survey.query.filter(Survey.researcher==g.user).order_by(Survey.created.
7                               desc())
8     return render_template('/researcher/index.html',
9                               tittle = 'Survey',
9                               surveys = surveys)
```

Figure 7.2: Ejemplo del controlador index

```
1 class SurveyForm(Form):
2     title = TextField('Title', validators = [Length(min = 1, max = 128)])
3     description = PageDownField('Description',validators = [Length(min = 0)],default
4                               = EXAMPLE_MARKDOWN)
5     startDate = DateTimeField('Day and start time', validators = [Optional()])
6     endDate = DateTimeField('Day and finish time', validators = [Optional()])
7     maxNumberRespondents = IntegerField('Number of respondents', validators = [
8         Optional()])
9     duration = IntegerField('Time in minutes that a user has to answer the survey',
10                             validators = [Optional()])
```

Figure 7.3: ejemplo de formulario

Si nos centramos mas en la función `index`, ver figura 7.2, cuando un manager accediese a la ruta `/researcher/index`, el servidor ejecutará el controlador `index`, que no recibe ningún parametro, y extrae todas las encuestas de un investigador. Estas encuestas se almacenan en una variable, *surveys*, y pasa ésta información como parametro a la función *render_template*, la cual se encarga de generar una página html a través de la plantilla y la variable pasada, para luego generar una respuesta HTTP con la página generada

7.2. Formularios

7.2.1. Creando un formulario

Para el uso de formularios en Flask, se ha hecho uso de Flask-WTF, que facilita la integración de WTForms en aplicaciones Flask. Todos los formularios se han declarado en en los ficheros *forms.py*.

Por ejemplo en la figura 7.2, se puede ver uno de los formularios que hay para la creación y la modificación de encuesta, en el cual el investigador puede definir las características generales de la encuesta, como puede ser el titulo, descripción, fecha de inicio, etc. Además de las restricciones que deben cumplir, como puede ser la longitud del texto.

```

1 def generate_form(questions):
2     '''dynamically generates the forms for surveys
3     '''
4     class AnswerForm(Form):
5         time = HiddenField('time',default=0)
6
7         for question in questions:
8
9             if isinstance (question,QuestionYN):
10                 choices = [('Yes',gettext('Yes')),('No',gettext('No'))]
11                 if question.isSubquestion:
12                     setattr(AnswerForm,str(question.id),MyRadioField('Answer',
13                     choices = choices,validators = [CheckSubquestion()])))
14                 ...
15             ...
16         form = AnswerForm()
17         return form

```

Figure 7.4: Generación dinámica de formularios

7.2.2. Creando un formulario dinámico

Muy raramente se necesita crear o modificar de manera dinámica los formularios usados. Pero debido a la naturaleza de nuestra aplicación, la cual permite la generación personalizada de encuestas, es necesario crear dinámicamente los formularios de las encuestas. Para ello se lee las preguntas de una sección de la encuesta y según las distintas propiedades de la pregunta se van añadiendo nuevos campos, que no son mas que nuevos atributos a la clase que forma el formulario, mediante la función *setattr*, como se puede ver en la figura 7.4

7.2.3. Creando un campo personalizado

WTForms provee una serie de campos básicos para la creación de formularios, además permite la personalización y creación de nuevos campos, para manejar tipos de datos especiales en la aplicación.

En el caso de la aplicación ha sido necesario crear nuevos campos personalizados para mostrar según que tipo de preguntas al usuario, como puede ser la escala likert que se puede ver en la figura 7.5. Para ello se ha partido de un campo ya existente, como es el *RadioField*, y se ha modificado la generación de código html.

Pudiéndose ver en la figura 7.6 el resultado final.

7.2.4. Creando un validador personalizado

WTForms provee una serie de validadores básicos, que permite verificar que la entrada de un campo cumple con algún criterio específico, como puede ser la longitud máxima de una cadena. Si la validación falla, lanza un *ValidationError*. Este es

7. CONTROLADOR

```
1 class LikertField(RadioField):
2     '''my implement of likert field'''
3     def __init__(self, label='', validators=None, labelMin="", labelMax="", **kwargs)
4         :
5         self.labelMin=labelMin
6         self.labelMax=labelMax
7         super(LikertField, self).__init__(label, validators, **kwargs)
8
9     def __call__(self, **kwargs):
10         '''render likert as table'''
11         from wtforms.widgets.core import html_params, HTMLString
12         kwargs.setdefault('id', self.id)
13         kwargs.setdefault('class_', " table table-condensed likert")
14         html = ['<%s %s>' % ("table", html_params(**kwargs))]
15         html.append('<tr>')
16         html.append('<td></td>')
17         for subfield in self:
18             html.append('<td>%s</td>' % (subfield.label))
19         html.append('</tr>')
20         html.append('<tr>')
21         html.append('<td class="type-info">%s</td>' % (self.labelMin))
22         for subfield in self:
23             html.append('<td>%s</td>' % (subfield()))
24         html.append('<td class="type-info">%s</td>' % (self.labelMax))
25         html.append('</tr>')
26
27         html.append('</%s>' % "table")
28         return HTMLString(''.join(html))
```

Figure 7.5: Campo likert

	1	2	3	4	5	6	7	
Desacuerdo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Totalmente acuerdo

Figure 7.6: Visualización de la escala likert


```

1 class CheckSubquestion(object):
2     '''check whether to answer the question or not'''
3
4     def __call__(self, form, field):
5         question = Question.query.get(field.name[1:])
6         # get answer of the question that depend
7         data = form["c"+str(question.parent.id)].data
8         if question.check_subquestion(data):
9             pass
10        else:
11            # nothing to check
12            field.errors[:] = []
13            raise StopValidation()

```

Figure 7.7: Validador personalizado

un sistema sencillo y flexible que permite añadir a cualquier campo una cadena de validadores.

Además de unos validadores básicos, WTForms permite la creación de validadores personalizados, que se adapten a las necesidades del sistema. Para ello

Al igual que WTForms provee una serie de validadores básicos, como puede ser *Lenght*, para comprobar la longitud de un campo introducido, también permite crear validadores personalizados, como se puede ver en la figura 7.7, se ha creado un validador para comprobar si una subpregunta hay que validarla o no, en función de la respuesta de la pregunta de la que depende.

7.2.5. Trabajo con formularios

Al principio de este capítulo hemos explicado el funcionamiento de un controlador, y durante el resto del capítulo se ha explicado la creación y personalización de formularios.

En la figura 7.4 se muestra un fragmento del controlador para editar encuestas, para ejemplificar como trabajan juntos el controlador y los formularios.

Cuando se recibe una petición para editar una encuesta, se busca dicha encuesta en el modelo *Survey.query.get(id_survey)*, y se utiliza los datos de la encuesta para crear una instancia del formulario, *form.title.data = survey.title*, esto permitirá que el formulario aparezca relleno con los valores actuales de la encuesta que se pretende editar.

Una vez que el investigador modifica el formulario y guarda los cambios, vuelve a entrar en juego el controlador, esta vez atendiendo a la petición POST. Se comprueba que los datos del formulario son validos mediante el metodo *form.validate_on_submit()*, si son validos se modifica el modelo y se guarda los cambios realizados, *survey.title*

7. CONTROLADOR

```
1 @blueprint.route('/survey/<int:id_survey>', methods = ['GET', 'POST'])
2 @login_required
3 @researcher_required
4 @belong_researcher('survey')
5 def editSurvey(id_survey):
6     survey = Survey.query.get(id_survey)
7     form = SurveyForm()
8     if form.validate_on_submit():
9         survey.title = form.title.data
10        ...
11        db.session.add(survey)
12        db.session.commit()
13        flash('Your changes have been saved.')
14    elif request.method != "POST":
15        form.title.data = survey.title
16        ...
17    return render_template('/researcher/editSurvey.html',
18        title = survey.title,
19        form = form,
20        survey = survey)
```

Figure 7.8: Controlador para editar encuestas

= *form.title.data*, *db.session.commit*. Si existiera algún error, al volver a generar la vista, se mostraría estos errores, *render_template*.

Chapter 8

Plantillas

En este capítulo se tratará de explicar como se generan las plantillas, que es la parte encargada de generar la vista en el patrón modelo vista controlador.

8.1. Motor de plantillas Jinja2

Jinja2 es un motor de plantillas, es decir un software que está diseñado para combinar una o mas plantillas con un modelo de datos para producir un documento resultado, en nuestro caso una página web.

Por otra parte cada plantilla contiene un esqueleto de la vista y una serie de variables y expresiones que son remplazadas con los valores obtenidos del modelo cuando se evalúa la plantilla para así poder generar una vista de manera dinámica.

En *Jinja2* existen dos tipos de delimitadores:

- `{% ...%}` es usado para ejecutar sentencias como puede ser bucles o la asignación de valores
- `{{..}}` se utiliza para imprimir variables o el resultado de una expresión.

Por ejemplo en el controlador *index* del modulo Researcher, se encarga de recoger de la base de datos todas las encuestas creadas por un investigador, estas encuestas son pasadas junto con una plantilla a la función `render_template` que se encarga de generar la vista. Esto se puede observar en la figura 8.1

Desde la plantilla podremos crear un listado con enlaces a las distintas encuestas del investigador, como se aprecia en la figura 8.2

8. PLANTILLAS

```
1 def index():
2     surveys = Survey.query.filter(Survey.researcher==g.user).order_by(Survey.created.
3         desc())
4     return render_template('/researcher/index.html',
5         tittle = 'Survey',
6         surveys = surveys)
```

Figure 8.1: Controlador researcher.index

```
1 {% extends "base.html" %}
2 {% block content %}
3     <h1>Researcher!</h1>
4     <div>
5         <a class="btn btn-primary btn" href="{{ url_for('researcher.new') }}">{{ ( '
6             New survey' ) }}</a>
7     </div>
8     <ul>
9         {% for survey in surveys %}
10             <li><a href="{{ url_for('researcher.editSurvey', id_survey = survey.id)
11                 }}">{{ survey.title }}</a></li>
12         {% endfor %}
13     </ul>
14 {% endblock %}
```

Figure 8.2: Plantilla de listado de encuestas

8.2. Estructura

El motor *Jinja2*, entre sus distintas características permite la herencia de plantillas, esto permite crear una plantilla base, que contiene todos los elementos comunes del sitio web y define los bloques que las plantillas descendientes pueden sustituir.

Como se puede ver en la figura 8.2, se ha optado por una plantilla base, `{% extends "base.html" %}`, que contiene los elementos básicos que se repiten a lo largo de las páginas web, como puede ser la barra de navegación o los ficheros de estilo y JavaScript.

Además para simplificar el uso de formularios en la página web se ha creado una serie de macros, que facilitan la generación del código de los formularios. Además se encargan de mostrar los errores si los ha habido a la hora de rellenar el formulario. Por ejemplo en la figura 8.3, se observa que se usa la macro *render_field*, para renderizar los distintos campos del formulario, así como indicar el tamaño que deben tener, o alguna *miga de pan*, con la información que debe incluir el investigador. En la figura 8.4 podemos ver el aspecto final una vez generado la plantilla. La figura 8.5 muestra un error producido a la hora de validar la plantilla.

```

1  {% from "formHelpers.html" import render_field %}
2  ...
3  <form method="post" class="form-horizontal">
4      {{form.hidden_tag()}}
5      {{ render_field(form.title, size = 64) }}
6      {{ render_field(form.description, rows = 10, style = 'width:100%') }}
7      {{ render_field(form.startDate, size = 128,placeholder="%Y-%m-%d %H:%M:%S") }}
8      {{ render_field(form.endDate, size = 128,placeholder="%Y-%m-%d %H:%M:%S") }}
9      {{ render_field(form.maxNumberRespondents) }}
10     {{ render_field(form.duration) }}
11     {{ render_field(form.surveyXml) }}
12     <div class="btn-group">
13         <input class="btn" type="submit" value="Create Survey">
14         <a class="btn" href="{% url_for('researcher.index') %}">{{ ('Cancel') }}</
15         a>
16     </div>
17 </form>

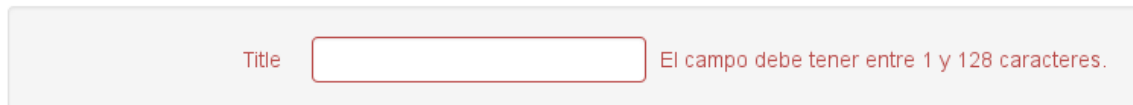
```

Figure 8.3: Plantilla nueva encuesta

The screenshot shows a web form for creating a new survey. The form is titled "Title" and has a text input field. Below it is a "Description" field with a rich text editor. The description field contains the text: "## This is a example of Markdown", "**Markdown** is rendered on the fly in the <i>preview area</i>", and "More about [markdown](http://daringfireball.net/projects/markdown/).". Below the description field is a preview area showing the rendered Markdown: "This is a example of Markdown", "Markdown is rendered on the fly in the preview area!", and "More about markdown.". Below the preview area are four input fields: "Day and start time" with a date and time format "%Y-%m-%d %H:%M:%S", "Day and finish time" with a date and time format "%Y-%m-%d %H:%M:%S", "Number of respondents" with a text input field, and "Time in minutes that a user has to answer the survey" with a text input field. Below these fields is a "File survey xml" field with a button "Examinar..." and the text "No se ha seleccionado ningún archivo.". At the bottom of the form are two buttons: "Create Survey" and "Cancel".

Figure 8.4: Vista generada de una nueva encuesta

8. PLANTILLAS



Title El campo debe tener entre 1 y 128 caracteres.

Figure 8.5: Vista generada de una nueva encuesta en la que hay un error

```
1  LANGUAGES = {  
2      'en': 'English',  
3      'es': 'Español'  
4  }
```

Figure 8.6: Lenguajes disponibles

8.3. Traducción

Para hacer que nuestra aplicación sea accesible a usuarios con distintos idiomas, se ha utilizado la extensión *Flask-Babel*, que proporciona una herramienta fácil de usar para traducir las distintas partes de la aplicación ya sea en el lado de las plantillas o ficheros escritos en python, todo ello usando la interfaz de internacionalización *gettext* de GNU.

8.3.1. Selección de idioma

La figura 8.6 contiene parte del fichero de configuración, donde están definidos los lenguajes disponibles. Luengo en cada petición que se realiza a la plataforma, Babel se encarga de decidir cual es el idioma a mostrar al usuario, como se puede ver en la figura 8.7.

8.3.2. Traducción de ficheros

Una vez que se tiene seleccionado el idioma, *Babel* se encarga de buscar una traducción a aquellos textos que se hayan señalado previamente. En los ficheros python se usa la función *gettext* para extraer el texto a traducir, como se puede observar en el siguiente ejemplo:

```
1 password = PasswordField(gettext('Password'), validators=[Required()])
```

```
1 from config import LANGUAGES  
2 @babel.localeselector  
3 def get_locale():  
4     return request.accept_languages.best_match(LANGUAGES.keys())
```

Figure 8.7: Selección de idioma

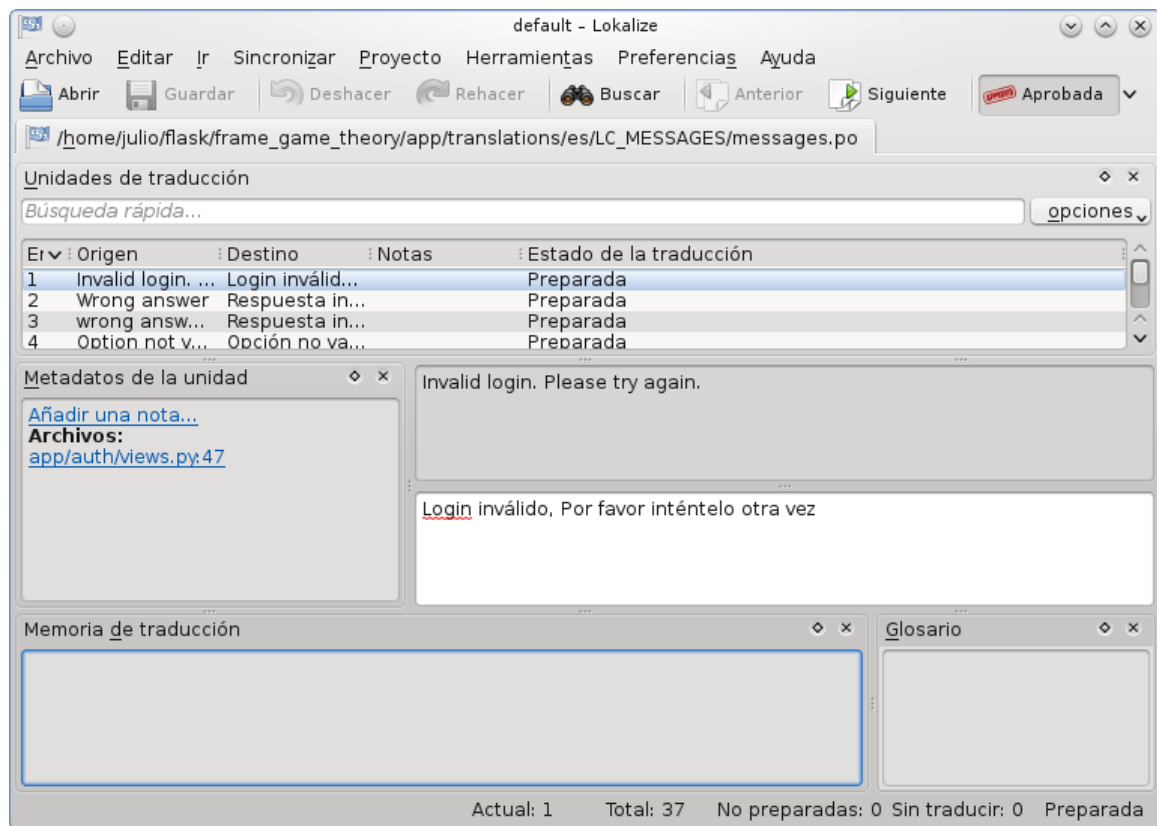


Figure 8.8: Captura de la herramienta Lokalize

Para la traducción de plantillas se realiza algo similar, pero en vez de *gettext*, se usa *_()*. Por ejemplo para traducir el enlace *Logout* de nuestra plantilla base:

```
1 <a href="{ url_for('auth.logout') }">{ _('Logout') }</a>
```

Una vez que tenemos seleccionado todos los textos a traducir de nuestras plantillas y ficheros python, se extraen todos ellos mediante el siguiente comando:

```
1 pybabel init -i messages.pot -d app/translations -l es
```

esto genera un catálogo con los textos a traducir, luego con una herramienta externa como puede ser Lokalize, ver figura 8.8, se traducen los textos a los distintos idiomas disponibles en la plataforma.

Por último, una vez traducido todos los textos hay que generar un fichero con los textos traducidos en un formato optimizado, para ello se utiliza el siguiente comando:

```
1 pybabel compile -d app/translations
```

8.4. JavaScripts

JavaScripts es un lenguaje de programación interpretado que se usa principalmente en el lado del cliente, implementado como una parte del navegador web, permitiendo entre otras cosas mejoras en la interfaz del usuario, comunicación asíncrona con el servidor o el añadir ciertas funcionalidades en el lado del cliente.

En las diferentes plantillas que generan código html, se ha incluido un bloque para los distintos códigos JavaScripts embebidos, que añaden distintas funcionalidades en la página web servida al usuario. Algunas de estas funcionalidades son las siguientes:

- En las vistas que muestran las distintas tablas con los resultados obtenidos en la encuesta así como en los distintos juegos, se puede ordenar la tabla por cualquier columna, evitando el tener que recargar la página y sobrecargar al servidor con esta operación.
- A la hora de crear o modificar encuestas y preguntas, la interfaz de usuario se ha mejorado y simplificado mostrando únicamente las distintas opciones posibles, así como ocultando las opciones menos comunes.
- Renderizado en tiempo real de los textos escritos en sintaxis Markdown.
- Generación de gráficos.
- Cálculo de tiempos empleados a la hora de contestar cada pregunta.
- Ocultar o mostrar las distintas preguntas dependiendo de las respuestas anteriores dadas.

Chapter 9

Pruebas

La fase de pruebas permite comprobar la corrección de los algoritmos de la aplicación y que cada módulo cumple con los requisitos iniciales planteados.

Cada apartado de este punto se refiere a un tipo de prueba realizado, como son los tests unitarios, los de carga y los de sistema, además de haber realizado varios tests con personas físicas.

Debido al ciclo de vida del proyecto, explicado en el apartado Gestión de tiempo, el desarrollo ha sido un ciclo de vida evolutivo, por lo que la automatización de pruebas se realizó al final del proyecto, y no en cada refinamiento del proyecto.

9.1. Tests unitarios y cobertura de código

En estos tests se ha querido comprobar el perfecto funcionamiento de todas las clases que se mapean en la base de datos, *models*, así como el correcto funcionamiento de los validadores creados para la correcta comprobación de los formularios, *forms* y la clase encargada en los emparejamientos de las decisiones, *game*.

Dentro de las pruebas unitarias se realizó pruebas de caja negra, que sirven para verificar que el item que se está probando, cuando se dan las entradas apropiadas produce los resultados esperados. Siendo este tipo de pruebas interesantes para comprobar el funcionamiento de las interfaces.

Además dentro de las pruebas unitarias, se realizó la cobertura de código, para comprobar el grado en que el código fuente del programa ha sido testeado en estas pruebas unitarias, así como la identificación de código nunca ejecutado.

Todos estos tests se integraron en la plataforma, pudiéndose ejecutar desde ella, como se puede ver en la figura 9.1

9. PRUEBAS

1	\$./manage.py test						
2	test_answer (test_models.ModelsTestCase) ... ok						
3	test_game_dictador (test_models.ModelsTestCase) ... ok						
4	...						
5	test_requerid_selectField (test_validators.ValidatorTest) ... ok						
6							
7	Module	statements	missing	excluded	branches	partial	
	coverage						
8	app/game/game	289	73	0	96	16	74%
9	app/main/views	14	13	0	0	0	7%
10	app/models	756	60	0	157	5	97%
11	app/surveys/forms	166	94	4	85	10	48%
12	app/surveys/utiles	42	23	0	22	7	44%
13	Total	1267	263	4	360	38	83%

Figure 9.1: Resultado de la ejecución de los test unitarios

Pese a no llegar a una cobertura cercana al 100%, si se analiza mas detalladamente los ficheros, se puede comprobar que si que se ha testeado las partes que se pretendía analizar.

9.2. Pruebas de carga

Este tipo de pruebas sirve para comprobar el rendimiento de la aplicación ante una carga de trabajo dado, con perspectiva para determinar el comportamiento de la plataforma web ante un acceso concurrente de cierto número de usuarios.

Para realizar el acceso concurrente, se utilizo la herramienta JMeter, que puede realizar todas las conexiones TCP que realizaría un usuario para contestar a la encuesta, además posee de un monitor para medir entre otros datos, los tiempos de respuesta.

9.3. Pruebas de Sistema

Para poder comprobar el correcto funcionamiento de la plataforma web, se carga una encuesta compleja, concretamente la del experimento "*¿Cómo son nuestros voluntarios?*".

Una vez cargado dicha encuesta, se utiliza la herramienta JMeter, para simular las acciones que realiza un voluntario al contestar una encuesta, cometiendo todo tipo de errores posibles y asegurandonos que los datos recibidos son los esperados.

Además de la automatización de esta prueba con JMeter, distintas personas usaron a su antojo la plataforma web y completaron el experimento para comprobar su correcto funcionamiento.

Chapter 10

Gestión del proyecto

En este capítulo se va a detallar la metodología usada para la gestión del proyecto.

10.1. Ciclo de vida del desarrollo.

A pesar de tener claro desde el principio cual era la meta del proyecto y del experimento "*¿Cómo son nuestros voluntarios?*", estos dos han ido evolucionado a la par durante el desarrollo del proyecto. Esto se ha debido sobretodo a las diferencias que hay a la hora de realizar una encuesta con apoyo presencial y a otra telemática.

Por eso el ciclo de vida que ha seguido el proyecto ha sido el prototipado evolutivo, como se puede ver en la figura 10.1, teniendo desde el primer mes una versión básica con los requisitos sacados del documento "*¿Cómo son nuestros voluntarios?*" y a partir de esta primera versión, con la información recibida por los investigadores, se han ido añadiendo características y refinando el funcionamiento de esta, así sucesivamente durante el desarrollo de la plataforma y la adaptación del experimento a esta.

Para facilitar esta retroalimentación de información, se ha intentado tener siempre una versión actualizada de la plataforma disponible para los investigadores y recibiendo los comentarios de estos a través del correo electrónico.

10.2. Gestión del tiempo

Debido al ciclo de vida desarrollado usado, una vez finalizado el primer prototipo, se ha vuelto constantemente a las etapas anteriores, añadiendo nuevos requisitos. Los grupos de tareas son los siguientes:

- **Análisis:** Se realizó un análisis previo al documento "*¿Cómo son nuestros voluntarios?*" y a partir de este y después de las sucesivas evaluaciones por

10. GESTIÓN DEL PROYECTO

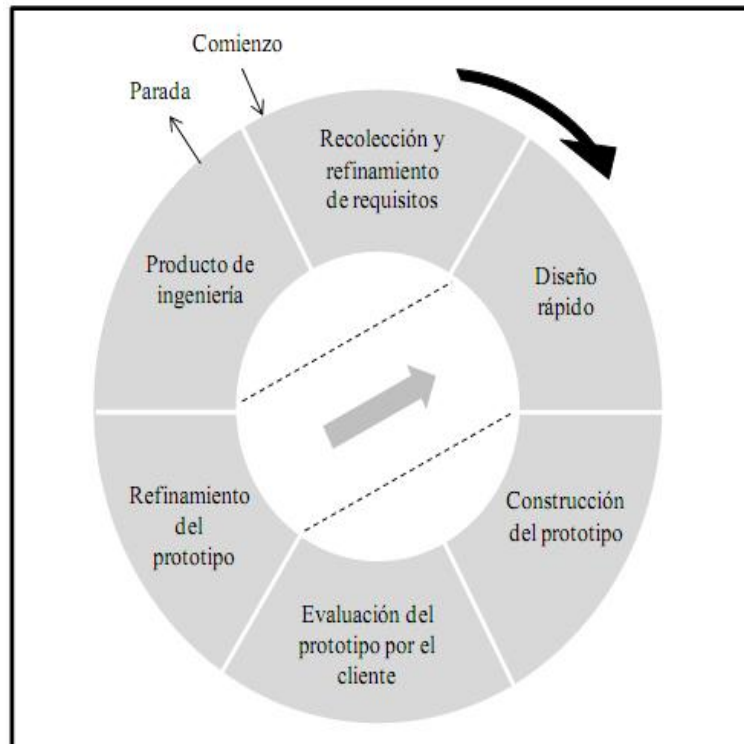


Figure 10.1: Esquema del prototipado evolutivo

parte de los investigadores de los distintos prototipos, se fueron añadiendo mas características.

- **Estudio de alternativas:** En este grupo se engloba el estudio de las distintas plataformas para la elaboración de encuestas, y su posible uso.
- **Estudio y familiarización de las herramientas:** Esto incluye el estudio de las diferentes tecnologías usadas para la elaboración del proyecto. A pesar de dedicar un tiempo en exclusiva para él, a lo largo del desarrollo se han ido adquiriendo nuevos conocimientos, que aveces ha implicado la reimplementación de ciertas funcionalidades a favor de un código y una solución mas clara.
- **Diseño:** Este grupo incluye desde la estructuración de la aplicación en una jerarquía de módulos y funcionalidades según el ámbito, hasta el diseño de los datos, así como el diseño del interfaz. En el diagrama de Gantt de la figura 10.2 solo se hace referencia explícita al primer diseño previo, pero después de cada evaluación se volvía al diseño.
- **Implementación:** Este grupo comprende en la escritura del código necesario para la elaboración de la plataforma.

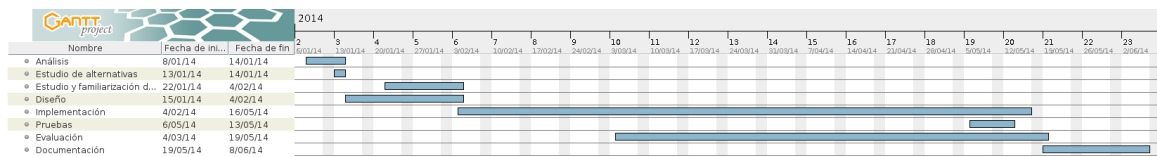


Figure 10.2: diagrama de Gantt

- **Pruebas:** Esto incluye las pruebas tanto unitarias, aceptación y de sobrecarga realizadas a la plataforma.
- **Documentación:** En esta parte solo se engloba la generación de la documentación para la elaboración de la memoria, aunque durante todas las fases se han ido generando distintos diagramas, y ficheros.
- **Evaluación:** En este grupo se incluye las distintas revisiones que han realizado los investigadores a la plataforma y al experimento "¿Cómo son nuestros voluntarios?". A lo largo de la vida del proyecto se ha ido recibiendo de manera constante retroalimentación por parte de los investigadores, que han ido refinando el experimento. Aun así se podría destacarse tres grandes revisiones en las siguientes fechas:
 - 4 de marzo, se enseñó por primera vez en Madrid una primera versión de la plataforma y del experimento. Además se decidió que el usuario podría recibir un feedback por parte del experimento.
 - 11 de abril, se revisó por completo el experimento, se recibió las pautas para modificarlo, además de indicar el formato de los ficheros de estadísticas para los investigadores.
 - 19 de mayo, se realizó la evaluación final de las características de la plataforma.

10. GESTIÓN DEL PROYECTO

Chapter 11

Conclusiones

Este capítulo contiene las conclusiones extraídas tras la realización de este Proyecto Final de Carrera y se apuntan algunas posibilidades de mejora y trabajo futuro. Se termina con una pequeña conclusión a nivel personal

11.1. Conclusiones generales

Una vez finalizado el proyecto, se puede concluir que los resultados obtenidos son satisfactorios, habiéndose cubierto el listado de los requisitos. Dichos requisitos se han ido ampliando a lo largo del proyecto, conforme iba evolucionando la plataforma, y el experimento "¿Cómo son nuestros voluntarios?", adaptando las necesidades de los investigadores a la plataforma.

Respecto al experimento "¿Cómo son nuestros voluntarios?", se llevará a cabo durante el mes de Julio, por lo que aun no podemos sacar conclusiones del comportamiento de la plataforma en un caso real, pero las diversas pruebas realizadas con gente ajena al experimento que ha probado la plataforma, así como los tests unitarios y pruebas de carga, parece indicar que no habrá problemas.

Además del desarrollo de la plataforma, se ha buscado la reutilización y ampliación de ésta, objetivo cumplido como se puede ver en el anexo ****Guía de desarrollo**** y la facilidad que tiene la plataforma para su ampliación.

11.2. Trabajo futuro

Durante la elaboración del proyecto siempre se ha tenido presente en crear un núcleo de la plataforma versátil y potente, el cual se pueda adaptar a las necesidades de otros proyectos. A continuación se proponen unas posibles mejoras, algunas de ellas completamente desarrolladas en el anexo ****Guía de desarrollo****:

11. CONCLUSIONES

- Añadir nuevos tipos de preguntas predeterminadas, que puedan servir para la elaboración de nuevas encuestas. Debido a la imposibilidad de llegar a todas las opciones posibles, durante el proyecto se decidió implementar solo el tipo de preguntas que aparecen en la encuesta "¿Cómo son nuestros voluntarios?", pero no por ello se quería dejar cerrada la plataforma a ese tipo de preguntas. Está fue una de las razones para escribir dos manuales en el anexo ****Guía de desarrollo****, de como incluir preguntas de tipo fecha y multitest.
- Aunque la interfaz de la plataforma esta traducida al Español e Ingles, y se puede ampliar fácilmente los idiomas soportados, no tiene soporte multiidioma, por lo que puede ser un inconveniente si se quiere acceder a poblaciones con distinto idioma. En el anexo ****Guía de desarrollo****, se dan las pautas necesarias para la implementar esta función.
- Crear un lenguaje específico del dominio. Durante el desarrollo del experimento "¿Cómo son nuestros voluntarios?", se observó que la creación de encuestas largas y complejas es una tarea tediosa y que consume bastante tiempo. Observando las soluciones de otras plataformas de encuestas se puede observar que ninguna es capaz de resolver este problema mediante el uso de una interfaz visual, ya sean mas o menos elaboradas y complejas. Por lo que se propone crear un lenguaje específico del dominio, como hace la plataforma ****Surveyor****. Una solución intermedia, pasa por la creación y modificación de la encuesta mediante ficheros XML, que ya soporta la plataforma o ficheros JSON, que también sería muy fácil de implementar. Esta última solución se ha probado con éxito en el experimento "¿Cómo son nuestros voluntarios?", sobre todo a la hora de modificar la encuesta.
- Creación de una API para extender la funcionalidad de la plataforma a otros dispositivos distintos del navegador web. Para la elaboración de esta tarea puede uno observar que funciones son las que tienen una vista para el navegador, y enviar y recibir los datos serializando estos mediante JSONS
- Desarrollo de un sistema de extensiones, para facilitar la ampliación de la plataforma, abstrayendo el núcleo de esta a la incorporación de nuevas funcionalidades.

11.3. Conclusión personal

La realización de este proyecto me ha servido entre otras cosas para ampliar mi conocimiento, trabajando en áreas en las que no se tenía conocimiento previo, como es el mundo de las aplicaciones web, y todas las tecnologías asociadas a esta. Aprendiendo y disfrutando mientras se realizaba el proyecto y pensando en lo que puede ser una nueva versión de la plataforma desarrollada.

Por otra parte se agradece la posibilidad de haber trabajado con algunos de los investigadores más importantes en su campo de investigación y de todo el feedback recibido por ellos, además del ambiente y la forma de trabajar en el Bifi, dentro de Ibercivis ha sido excelente.

También se ha valorado positivamente la libertad dada en cuanto al uso de tecnologías y el haber podido desarrollar la plataforma siempre de manera transparente y libre, mediante el uso de GitHub, pudiendo devolver algo a la comunidad de software libre de la que tanto se ha tomado de ella.

11. CONCLUSIONES

Appendix A

Manual de administrador

En esta sección vamos a explicar como un administrador puede instalar la aplicación y ponerla en marcha

A.1. Consideraciones

Aunque este manual intenta ser autoexplicativo en cuanto a la instalación de la plataforma, no es un manual de las distintas herramientas usadas en la plataforma, por lo que para ello se remite a la documentación oficial de estas. Por otra parte se espera que el administrador de la aplicación tenga nociones en la administración de servidores web, así como unas nociones muy básicas de Python.

A.2. Requisitos hardware

Los requisitos hardware dependerán del número de usuarios que van a acceder a la plataforma.

Si se quiere medir como se comporta la plataforma ante un número elevado de usuarios, en la carpeta `jmeter` se incluye una configuración para Apache JMeter, así como un generador de usuarios y una encuesta compleja. Esta aplicación genera todas las peticiones que realizaría un usuario al contestar a una encuesta. Para ejecutar la plataforma en este modo, debe cambiar la variable de entorno `FLASK_CONFIG` a `jmeterProduction` o en el fichero `config.py` cambiar la configuración default por `jmeterProduction`.

También existe la posibilidad de ejecutar la plataforma en la nube. Dependiendo de la elección llevará mas o menos cambio. En el anexo C se incluyen las modificaciones y los pasos necesarios para ejecutar la aplicación en Heroku.

A. MANUAL DE ADMINISTRADOR

```
1 config = {
2     'development': DevelopmentConfig,
3     'testing': TestingConfig,
4     'production': ProductionConfig,
5     'heroku': HerokuConfig,
6     'unix': UnixConfig,
7     'jmeter': Jmeter,
8     'jmeterProduction' : JmeterProduction,
9
10     'default': DevelopmentConfig }
11 \end{listing}
```

Figure A.1: Parte del fichero config.py

A.3. Requisitos software

Los requisitos software son los siguientes:

- Virtualenv: Para la creación de un entorno virtual de Python para la instalación de todas los módulos necesarios.
- Git: Para poder descargarse la aplicación.
- Servidor web compatible con WSGI, en la documentación de Flask puedes encontrar una guía rápida para la puesta en marcha del servidor, <http://flask.pocoo.org/docs/deployin>
- Base de datos a usar, está debe ser compatible con SQLAlchemy, las posibilidades son las siguientes:
 - Postgresql
 - MySQL y su fork MariaDB
 - Oracle * Microsoft SQL Server
 - SQLite

A.4. Instalación

Empezaremos clonando el repositorio git donde se encuentra la aplicación:

```
1 $ git clone git://github.com/nu_kru/swarm-survey.git
2 $ cd swarm-survey
```

Una vez clonado el repositorio procederemos a crear un entorno virtual para poder instalar todas las dependencias necesarias en la aplicación:

```
1 $ mkdir swarm-surveys
2 $ virtualenv venv
3 New python executable in venv/bin/python
4 Installing distribute.....done.
```

Una vez instalado el entorno virtual, para activarlo:

```
1 $ . venv/bin/activate
```

Una vez dentro del entorno virtual procederemos a instalar todas las dependencias de la aplicación. Todas ellas, así como la versión usada se encuentran en el fichero *requirements.txt*. Para instalarla haremos uso de pip, el cual se ha instalado dentro de *virtualenv*:

```
1 (venv)$ pip install -r requirements.txt
```

Con esto ya tendremos instalada la aplicación así como todos los módulos necesarios para hacerla funcionar.

A.5. Configuración

La configuración de la plataforma está localizada en los ficheros escritos en Python, *config.py* y *settings*.

En el fichero *config.py* se puede observar los distintos modos en los que se puede arrancar la aplicación, estos son los siguientes:

- **Development:** para el desarrollo de la plataforma.
- **Testing:** para ejecutar los test unitarios de la plataforma.
- **Production:** configuración base para el modo producción
- **Heroku:** para ejecutar la aplicación en modo producción en la nube Heroku.
- **Unix:** para ejecutar la aplicación en modo producción en una máquina de tipo Unix
- **Jmeter:** para ejecutar el test de sobrecargar y aceptación
- **JmeterProduction:** para ejecutar el test de sobrecarga y aceptación en una máquina en modo producción.

Para cambiar de modo, puede hacerlo mediante la variable de entorno *CONFIG_FLASK* o sino está definida modificando la configuración por defecto en el fichero *config.py*.

En el fichero *settings* se encuentra una plantilla con las variables a cambiar. La plataforma espera encontrar la localización del fichero usando la variable de entorno *SWARMS_SURVEY_SETTINGS* o sino por defecto el fichero *settings.cfg* el cual se genera automáticamente si no se encuentra.

A. MANUAL DE ADMINISTRADOR

Tanto el fichero *settings* como *config.py* son autoexplicativos, estando documentado el significado de cada opción. En el fichero *settings* básicamente hay que indicar cual es el servidor de correo y los usuarios a los cuales se le va a enviar los correos con las distintas alertas que puede generar la plataforma, así como el usuario administrador y la contraseña de este.

A.6. Base de datos

A.6.1. Configuración de la base de datos a usar

La plataforma hace uso de la variable de entorno `**DEV_DATABASE_URL**` en la cual espera que se encuentre la URI de la conexión de la base de datos, así como el usuario y contraseña. Si la variable no está definida usa por defecto SQLite.

El formato es el siguiente:

```
1 driver://username:password@host:port/database
```

Por ejemplo para MySQL:

```
1 driver://username:password@host:port/database
```

Mas información para la configuración de la base de datos consulte la documentación oficial de SQLAlchemy:

http://docs.sqlalchemy.org/en/rel_0_9/core/engines.html

A.6.2. Creación de la base de datos

Una vez definida la base de datos a usar entre las soportadas por SQLALchemy, debemos de crear la base de datos, para ello ejecutaremos los siguientes comandos:

```
1 (venv)$./manage.py db init
2 (venv)$./manage.py db migrate
3 (venv)$./manage.py db upgrade
```

Con esto generamos la base de datos, además se hace uso de *flask-migrate*, una extensión que hace uso de *Alembic* para poder migrar la base de datos a nuevas actualizaciones del sistema. Para obtener mas información de como migrar o volver a una revisión anterior de la base de datos, consulte la documentación oficial: <http://flask-migrate.readthedocs.org/en/latest/>

```
1 #Inicio de la shell:
2 (venv)$ manage.py shell
3 #Dar permisos de investigador al usuario foo@foo.com:
4 >>>add_researcher(foo@foo.com)
5 #Quitar permisos de investigador al usuario foo@foo.com
6 >>>delete_researcher(foo@foo.com)
7 #Listar usuarios disponibles:
8 >>>list_user()
```

Figure A.2: Ejemplos de comandos disponibles en la shell

A.7. Inicio del programa

Dependiendo del servidor web usado, deberá arrancar la aplicación de una manera u otra, para ello diríjase a la documentación oficial de su servidor web o consulte como guía rápida, <http://flask.pocoo.org/docs/deploying/>

```
1 #Usando el servidor Gunicorn:
2 (venv)$ gunicorn manage.py runserver:app
3 #Usando el servidor de desarrollo de Flask:
4 (venv)$ manage.py runserver
```

Una vez puesta en marcha la plataforma, se crea automáticamente en la base de datos el usuario administrador indicado en el fichero de configuración, además también se le otorgan roles de investigador para poder crear encuestas.

A.8. Asignación del rol investigador a un usuario

Para asignar el rol de investigador a un usuario, inicie la shell del programa. Esto abrirá un interprete Python de la plataforma web. Entre las funciones disponibles están la de *add_researcher* y *delete_researcher*, que sirven para dar y quitar permisos de investigador a un usuario dado. En la Figura A.2, hay una muestra de comandos disponibles.

A.9. Actualización

Antes de actualizar la plataforma se recomienda hacer una copia de seguridad de la base de datos y del entorno virtualizado donde se ha instalado la plataforma. Los pasos para actualizar son los siguientes:

- Haga una copia de la configuración de la plataforma, *config.py* y *settings.cfg*.
- Descargue la última versión de la plataforma mediante git:

```
1 (venv)$ git pull
```

A. MANUAL DE ADMINISTRADOR

- Instale los nuevos módulos requeridos:

```
1 (venv)$pip install -r requeriments.txt
```

- Compruebe si ha habido algún cambio en los ficheros de configuración.
- Actualice la base de datos si es necesario:

```
1 (venv)$./manage.py db migrate
2 (venv)$./manage.py db upgrade
```

A.10. Log

La plataforma por defecto guarda todos los mensajes con un nivel *warning* o superior en el fichero *temp/swarms.log* esta configuración se puede cambiar en el fichero *config.py*, también se puede elegir usar *SysLogHandler* para comunicarse remotamente con una maquina Unix, quién almacenará la información.

Además tambien se enviará por correo al usuario indicado los mensajes con un nivel *error* o superior.

El nivel de los mensajes se puede cambiar, los disponibles son los siguientes:

- critical.
- error.
- warning.
- info.
- debug.

Mas información disponible en <https://docs.python.org/2/library/logging.html>

Appendix B

Manual de investigador

En este manual vamos a tratar las opciones que tiene un investigador para crear una encuesta.

B.1. Inicio de sesión

Existen dos posibilidades para iniciar la sesión, mediante el uso de una cuenta OpenID para lo cual puedes usar su cuenta de Google, Yahoo, Steam o cualquier otro servicio que haga uso de OpenID. O mediante el uso de un usuario y contraseña. Para ello previamente debe registrarse en la plataforma, y se le enviará un correo con la contraseña.

B.1.1. OpenID

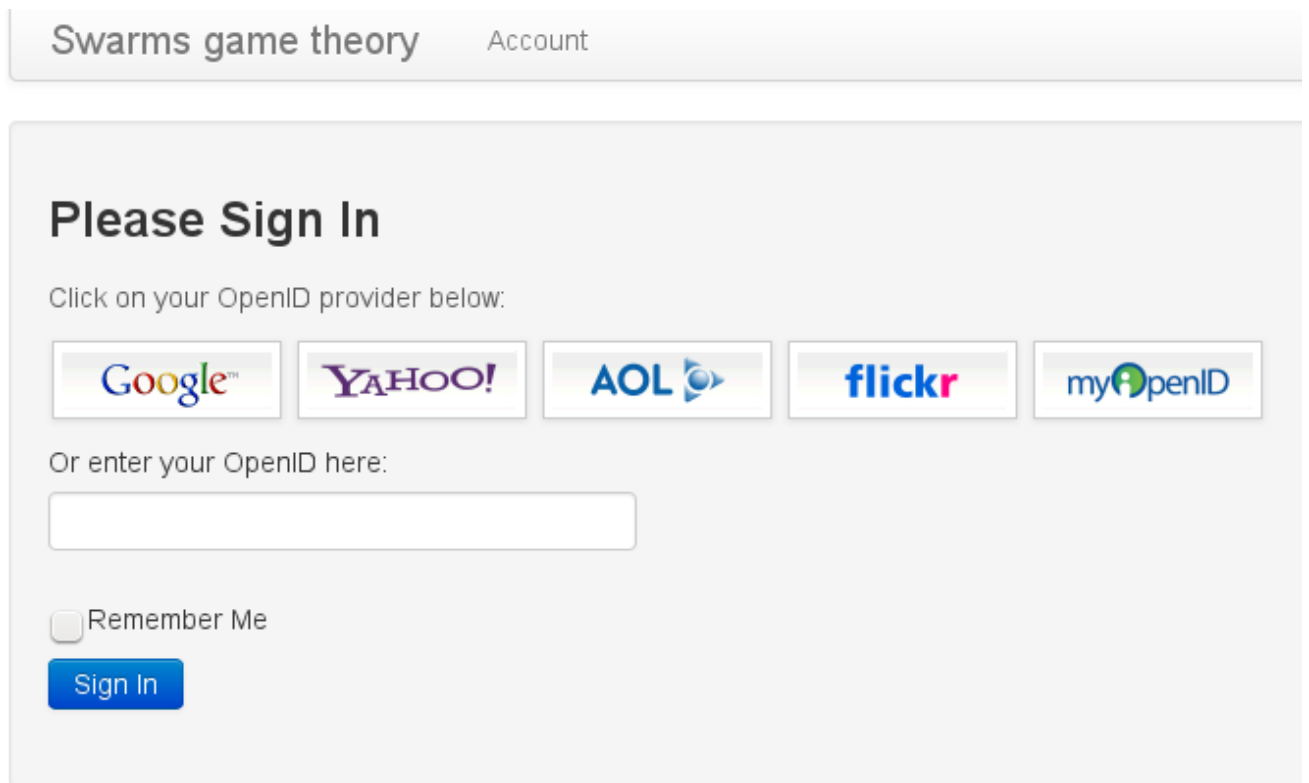
Haciendo uso de un navegador entre en el portal web donde se encuentra alojada la plataforma y entre en el enlace "Account", ver figura B.1



Indice! Información

Esta web es solo una muestra de la plataforma que esta en desarrollo y puede no contener la última versión de desarrollo, la base de datos puede

Figure B.1: Página principal de la aplicación



Swarms game theory Account

Please Sign In

Click on your OpenID provider below:

Google™ YAHOO! AOL flickr myOpenID

Or enter your OpenID here:

☐ Remember Me

Sign In

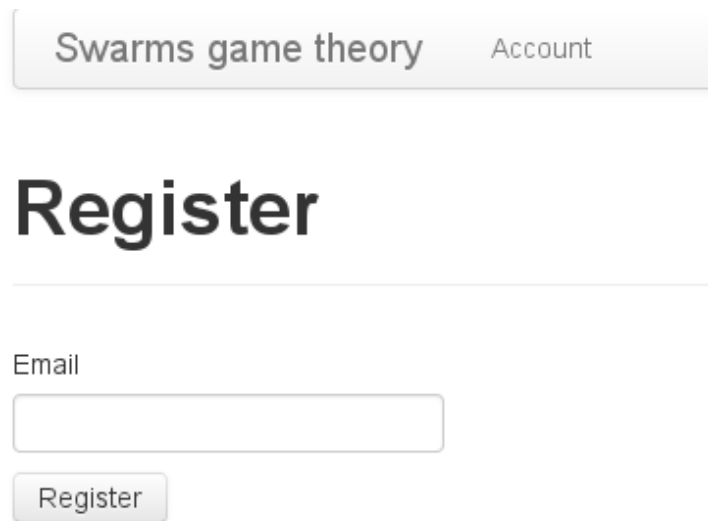
Figure B.2: Página de inicio de sesión mediante OpenID

Escriba la dirección del servidor OpenID el cual va usar para autenticarse o haga click sobre uno de los servidores predeterminados. Para entrar en la plataforma aprieta al botón Sign In, ver figura B.2

B.1.2. Correo

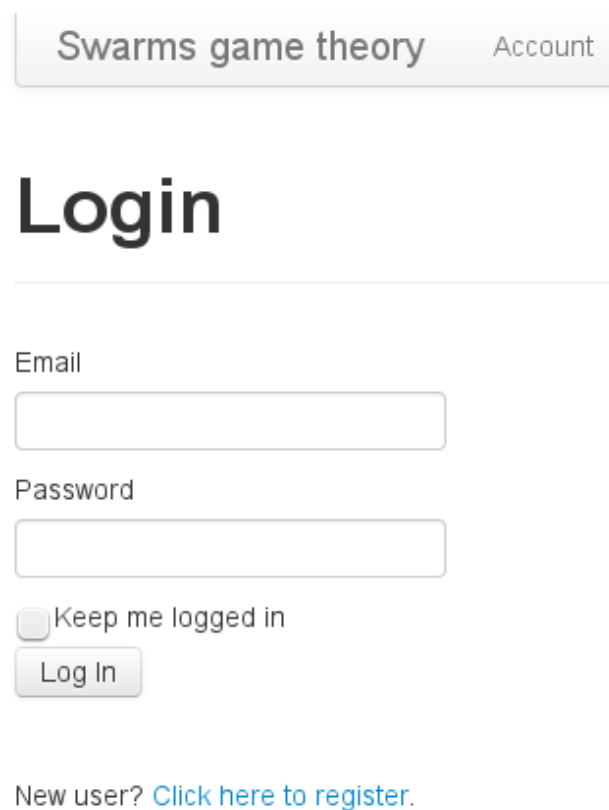
Para registrarse en la plataforma, vaya a la dirección <http://servidor/auth/register>, ver figura B.3, e introduzca el correo que quiere usar de registro. Una vez registrado, se le enviara un correo con la contraseña.

Para iniciar sesión, vaya a la dirección <http://servidor/auth/loginEmail> e introduzca el correo con la que se registró y la contraseña que se le envió al correo, ver figura B.4:



The screenshot shows the 'Register' page of a web application. At the top, there is a navigation bar with two links: 'Swarms game theory' and 'Account'. Below the navigation bar, the word 'Register' is displayed in a large, bold font. Underneath, there is a label 'Email' followed by a text input field. Below the input field is a button labeled 'Register'.

Figure B.3: Página de registro mediante correo



The screenshot shows the 'Login' page of a web application. At the top, there is a navigation bar with two links: 'Swarms game theory' and 'Account'. Below the navigation bar, the word 'Login' is displayed in a large, bold font. Underneath, there is a label 'Email' followed by a text input field. Below the email field is a label 'Password' followed by a text input field. Below the password field is a checkbox labeled 'Keep me logged in'. Below the checkbox is a button labeled 'Log In'. At the bottom of the page, there is a link that says 'New user? [Click here to register.](#)'.

Figure B.4: Página de inicio mediante correo/contraseña

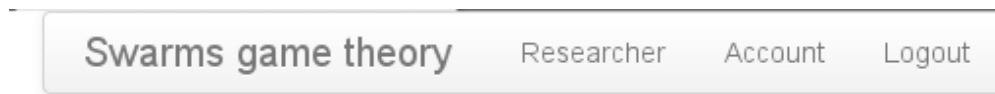


Figure B.5: Barra de navegación

B.2. Creación de encuestas

B.2.1. Crear o editar una encuesta

Una vez iniciado sesión en la plataforma, en la barra de navegación de la plataforma verá dos enlaces, figura B.5:

- **Logout:** para salir de la sesión.
- **Researcher:** para entrar en el módulo de creación de encuestas.

Una vez dentro del modulo de creación de encuestas observara una página similar a la figura B.6, las opciones disponibles son las siguientes:

1. Botón **New survey:** Opción para crear una nueva encuesta
2. Opción para modificar una encuesta ya creada, en este caso la encuesta *¿Cómo son nuestros voluntarios?*
3. Botón **export stats:** Opción para exportar los resultados de una encuesta

Si se entra en la opción de crear encuestas, verá en la figura B.7 los distintos campos a rellenar para crear una encuesta:

1. **Title:** Título de la encuesta, este campo es obligatorio.
2. **Description:** Descripción de la encuesta, esta descripción soporta sintaxis Markdown. Este campo es opcional
3. Previsualización de la descripción
4. **Day and start time:** Día y hora en la que comenzará la encuesta. Este campo es opcional, sino es rellenado no se mostrará la encuesta a los usuarios
5. **Day and finish time:** Día y hora en la que finalizará la encuesta. Este campo es opcional, sino es rellenado, la encuesta no tendrá fecha de finalización

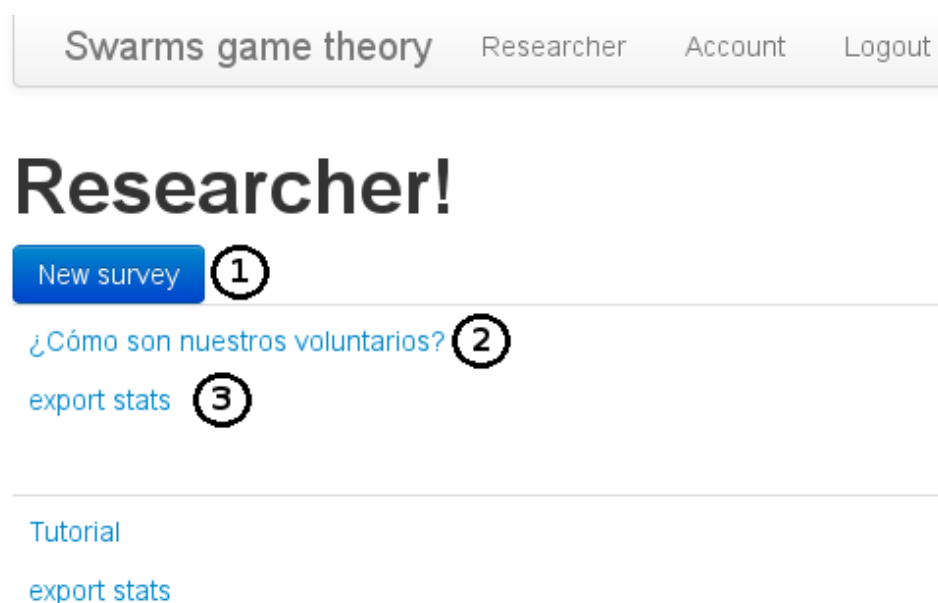


Figure B.6: Página de inicio del módulo de creación de encuestas

6. **Number of respondents:** Número máximo de usuarios que podrán responder a la encuesta. Este numero es orientativo, una vez llegado a ese número, no se permitirá a ningún usuario mas empezar la encuesta, pero si terminarla. Este campo es opcional, sino se rellena, no hay número máximo
7. **Time in minutes that a user has to answer the survey:** Número máximo en minutos que tiene un usuario para terminar la encuesta. Este campo es opcional, sino se rellena no hay tiempo máximo
8. **File survey xml:** Para importar una encuesta previamente exportada.
9. **Create survey:** Para crear la encuesta. Si hay algún error se le mostrará indicado cual es, como sucede en la figura B.8.

Una vez creada la encuesta, podrá modificar cualquier campo de está. No así las preguntas una vez haya sido publicada la encuesta. Después de crear la encuesta, verá la figura B.2.1 con las siguientes opciones:

1. **Save changes:** Guardar los cambios realizados en esta pantalla.
2. **View/Add consents:** Ver y añadir consentimientos a la encuesta.
3. **Add section:** Añadir una sección nueva a la encuesta.

Swarms game theory

Researcher

Account

Logout

New Survey

①

Title

②

Description

③

④

Day and start time

⑤

Day and finish time

⑥

Number of respondents

⑦

Time in minutes that a user has to answer the survey

⑧

File survey xml

⑨

Create Survey

Cancel

This is a example of Markdown

Markdown is rendered on the fly in the <i>preview area</i>!

More about [markdown](http://daringfireball.net/projects/markdown/).

This is a example of Markdown

Markdown is rendered on the fly in the *preview area*!

More about [markdown](#).

%Y-%m-%d %H:%M:%S

%Y-%m-%d %H:%M:%S

Examinar...

No se ha seleccionado ningún archivo.

Figure B.7: Página con los distintos campos de una encuesta

Swarms game theory

Researcher

Account

Logout

New Survey

Title

Field must be between 1 and 128 characters long.

Description

Day and start time

%Y-%m-%d %H:%M:%S

Day and finish time

%Y-%m-%d %H:%M:%S

Number of respondents

Time in minutes that a user has to answer the survey

diez

Not a valid integer value

File survey xml

Examinar...

No se ha seleccionado ningún archivo.

Create Survey

Cancel

Figure B.8: Error mostrado al crear una encuesta

B. MANUAL DE INVESTIGADOR

Swarms game theory Researcher Account Logout

Your survey have been saved.

Edit Survey

⑥

- Parte1
 - 1.1
 - 1.2

Title

Tutorial

Description

Encuesta Tutorial

Todo el texto hace uso de sintaxis [markdown](http://daringfireball.net/projects/markdown/).

Encuesta Tutorial

Todo el texto hace uso de sintaxis [markdown](#).

Day and start time

2014-08-01 14:29:01

Day and finish time

2014-10-01 14:29:01

Number of respondents

Time in minutes that a user has to answer the survey

①

②

③

④

⑤

Save changes

View/Add consents

Add section

Delete survey

Export survey

Figure B.9: Página de inicio de una encuesta ya creada

Figure B.10: Página para crear un nuevo consentimiento

4. **Delete survey:** Eliminar la encuesta.
5. **Export survey:** Exportar la encuesta a un fichero XML con la posibilidad mas tarde de importar la encuesta.
6. Arbol de secciones de la encuesta.

B.2.2. Añadir y editar consentimientos

Si realizada click en el botón de **View/Add consents**, figura B.2.1, irá a una pantalla con las opciones de añadir, editar y eliminar los consentimientos de una encuesta, figuras B.10 y B.11. El formato usado nuevamente para la escritura de texto es Markdown

B.2.3. Insertar una nueva sección

Si realiza click en el botón de **Add section**, figura B.2.1, irá a una pantalla, ver figura B.12, con las opciones para añadir una sección nueva, estas son las siguientes:

Swarms game theory

Researcher

Account

Logout

Adding consent.

Consent!

Consent

This is a example of Markdown
Markdown is rendered on the fly in the *<i>preview area</i>!*

More about [markdown](http://daringfireball.net/projects/markdown/).

This is a example of Markdown
Markdown is rendered on the fly in the *preview area*!
More about [markdown](#).

Add ConsentBack

Consentimiento 1

Delete Edit

Figure B.11: Página con un consentimiento ya creado

Swarms game theory

Researcher

Account

Logout

Tutorial

Section!

1

Title

Título

2

Description

Texto descriptivo

3

Sequence

1

4

Percent

1.00

5

Add Section

Back

Figure B.12: Página de nueva sección

B. MANUAL DE INVESTIGADOR

1. **Title:** Título de la sección, este campo es obligatorio.
2. **Description:** Descripción de la sección, esta descripción soporta sintaxis Mark-down. Este campo es opcional.
3. **Sequence:** Posición en la que aparecerá esta sección
4. **Percent:** Porcentaje de usuarios que realizarán esta sección.

Si dos o mas secciones del mismo nivel tienen la misma secuencia, se elegirá al azar el orden de las secciones con la misma secuencia. Este orden se calculará cada vez para cada usuario que empiece una encuesta. Por lo que un usuario puede realizar la encuesta con un orden distinto a otro usuario.

Las secciones del mismo nivel con misma secuencia y porcentaje distinto de 1 son excluyentes. Esto quiere decir que si tienes tres secciones: *Sección 1* con 0.3, *Sección 2* con 0.2 y *Sección 3* con 0.5. El 30% de usuarios realizarán la *Sección 1*, el 20% realizarán la *Sección 2* y el 50% realizarán la *Sección 3*.

Una vez creada una sección, ver figura B.13, se le mostrarán las opciones para eliminar una sección, añadir una subsección a la sección dada, añadir/editar preguntas y duplicar la sección.

B.2.4. Añadir nuevas preguntas

Si realiza click en el botón de **Add/Edit question** dentro de una sección, B.13, ira a la página para añadir/editar preguntas.

B.2.4.1. Preguntas cuya respuesta es Si o No

Las opciones para las preguntas **Si o No**, ver figura B.14, son las siguientes:

1. **Text:** Texto de la pregunta, esta descripción soporta sintaxis Markdown.
2. **Required:** Si es una pregunta obligatoria o no.
3. **Answer:** La respuesta correcta de la pregunta si es que la tiene. “Yes”, si la respuesta correcta es *si*, “No” si la respuesta correcta es no. Este campo es opcional.
4. **Number of attempt:** Número de intentos para responder a la pregunta, en caso que la pregunta tenga una respuesta correcta. Nada, para infinitos intentos.

El apartado **SubQuestion** y **Options game** se explicará mas adelante.

Swarms game theory Researcher Account Logout

Save changes

Tutorial

Parte1 /

Section!

- Parte1

Title

Parte1

Description

Texto descriptivo

Sequence

1

Percent

1.00

Save changes

Add subsection

Add/Edit question

Back

.Parte1

Texto descriptivo

Sequence: 1

Percent: 1.0

[Delete](#) [Add subsection](#) [Add/Edit question](#) [Duplicate section](#)

Figure B.13: Sección ya creada

Tutorial

Parte1 / 1.1 /

Add Question!!

- Parte1
 - 1.1
 - 1.2

①

Text

/projects/markdown/).

/projects/markdown/).

②

Required

☒

③

Type of question

YES/NO

④

answer

none if There isnt correct answer

④

Number of attempt

SubQuestion

Type of operation

None

Value

Question

Options game

Add Question

Figure B.14: Creación de preguntas de tipo Si y No

B.2.4.2. Preguntas cuya respuesta es un texto

Las opciones para las preguntas cuya respuesta es un texto, ver figura B.15, son las siguientes:

1. **Number:** Si se desea validar que el texto introducido por el usuario en un número entero.
2. **Number Float:** Si se desea validar que el texto introducido por el usuario en un número flotante.
3. **Regular Expression:** Si se desea validar que el texto introducido por el usuario corresponde a una expresión regular, para ello se usa la sintaxis de Python de expresiones regulares.
4. **Error Message:** Mensaje de error personalizado ante el fallo a la hora de introducir la respuesta el usuario.
5. **Answer:** La respuesta correcta de la pregunta si es que la tiene. Se comprueba si es correcta después de validar que la respuesta tenga el formato apropiado. A la hora de comprobar la respuesta se ignora la coincidencia de mayúsculas/minúsculas.
6. **Number of attempt:** Número de intentos para responder a la pregunta, en caso que la pregunta tenga una respuesta correcta. Nada, para infinitos intentos.

B.2.4.3. Preguntas cuya respuesta es una opción posible

Las opciones para las preguntas cuya respuesta es una opción posible, ver figura B.16, son las siguientes:

1. **Answer:** Respuesta correcta a la pregunta.
2. **Number of attempt:** Número de intentos para responder a la pregunta.
3. **Render:** Formato a la hora de mostrar las opciones posibles al usuario. Son tres:
 - **Vertical:** Se muestran todas las opciones en vertical
 - **Horizontal:** Se muestran todas las opciones en horizontal.

- [Parte 1](#)
 - [1.1](#)
 - [1.2](#)

Text

/projects/markdown/).

/projects/markdown/).

Required ☒

Type of question

Text

① Number ☐

② Number Float ☐

③ Regular Expression

④ Error Message

⑤ answer

none if There isnt correct answer

⑥ Number of attempt

SubQuestion

Type of operation

None

Value

Question

Options game

Add Question

Figure B.15: Creación de preguntas de tipo texto

- Parte1
 - 1.1
 - 1.2

Text

/projects/markdown/).

/projects/markdown/).

Required ☒

Type of question

Choice

1 answer

none if There isnt correct answer

2 Number of attempt

3 Render

vertical

4 Range step

1.00

5 Range min

6 Range max

7 Answer 1

8 Add other answer

SubQuestion

Type of operation

None

Value

Question

Options game

Add Question

Figure B.16: Creación de preguntas de tipo selección

B. MANUAL DE INVESTIGADOR

- **Select:** Se muestran todas las opciones en un formulario de selección como el elegido para mostrar las opciones de **Render**
- 4. **Range step:** Si la respuesta esta formado por un rango de números, indica el salto entre un número y el siguiente.
- 5. **Range min:** Rango de inicio, si la respuesta está formado por un rango de números.
- 6. **Range max:** Rango máximo, si la respuesta está formado por un rango de números.
- 7. **Answer 1:** Respuesta posible.
- 8. **Add other answer:** Añadir otra respuesta posible.

B.2.4.4. Preguntas cuya respuesta es una escala likert

Las opciones para las preguntas cuya respuesta es una escala likert, ver figura B.17, son las siguientes:

1. **Scale:** Número inicial de la escala likert, puede ser 0 o 1.
2. **to:** Número final de la escala likert.
3. **min:** Etiqueta opcional para indicar el significado del valor mínimo de la escala.
4. **max:** Etiqueta opcional para indicar el valor máximo de la escala.

B.2.4.5. Preguntas dependientes de la respuesta a otra pregunta

Todas las preguntas anteriores se pueden mostrar o no dependiendo de la respuesta dada a una pregunta. En la figura B.2.4.5 podemos ver las siguientes opciones:

1. **SubQuestion:** Menú desplegable que muestra las opciones para las subpreguntas
2. **Type of operation:** Operación con la que realizaremos la comparación, esta puede ser:
 - **Mayor,** >
 - **Menor,** <

- Parte1
 - 1.1
 - 1.2

Text

/projects/markdown/).

/projects/markdown/).

Required

☒

Type of question

Likert Scale

1

Scale

1

2

to

7

3

min

label optional

4

max

label optional

SubQuestion

Type of operation

None

Value

Question

Options game

Add Question

Figure B.17: Creación de preguntas de tipo escala likert

Swarms game theory

Researcher

Account

Logout

¿Cómo son nuestros voluntarios?

PRIMERA PARTE / Bloque 1 /

Add Question!!

- PRIMERA PARTE
 - Bloque 1
 - Bloque 2
 - Bloque 3
 - Bloque 4
 - Bloque 5
- SEGUNDA PARTE
 - 2. Dinero Real
 - Bloque 1
 - Bloque 2
 - 2. Dinero Ficticio
 - Bloque 1

Text

5.1. ¿Viven todos actualmente?:

5.1. ¿Viven todos actualmente?:

Required

☐

Type of question

YES/NO

answer

none if There isnt correct answer

Number of attempt

0

1 SubQuestion

2 Type of operation

>

3 Value

0

4 Question

5. ¿Cuántos hijos has teni

Figure B.18: Creación de preguntas dependientes de otras preguntas

5. ¿Cuántos hijos has tenido?

6. Consideras que tu estado de salud es:

- ☐ muy malo
- ☐ malo
- ☐ regular
- ☐ bueno
- ☐ muy bueno

Figure B.19: Ejemplo de pregunta dependiente

- **Igual**, =
- **Distinto**, !=

3. **Value**: Valor a comparar según la operación anterior indicada

4. **Question**: Pregunta de la que depende la respuesta

En las preguntas de Si o No, las respuestas posibles son "Yes" y "No" y solo tiene sentido las operaciones igual y distinto.

En las preguntas cuya respuesta es un texto, todas las operaciones son posibles.

En las preguntas de selección se muestra en **Value** todas las opciones posibles.

En las preguntas de escala likert, se muestra en **Value** todas las opciones posibles.

Las preguntas que dependen de otra respuesta solo se mostrará al usuario dependiendo de la respuesta dada, como se puede ver en las figuras B.19 y B.20.

B.2.4.6. Preguntas de Juegos

Estas opciones solo tienen sentido para la elaboración de encuestas en las que haya una serie de juegos económicos en los que participan los usuarios. Cada juego va ligado automáticamente a un tipo de pregunta, siendo imposible cambiarlo. En la figura B.21 podemos ver los juegos/preguntas disponibles, con las siguientes opciones:

- **Part two**: Es una serie de preguntas que mide la impaciencia del jugador.

5. ¿Cuántos hijos has tenido?

5.1. ¿Viven todos actualmente?:

☐ Yes

☐ No

6. Consideras que tu estado de salud es:

☐ muy malo

☐ malo

☐ regular

☐ bueno

☐ muy bueno

Figure B.20: Ejemplo de pregunta dependiente

- [Parte1](#)
 - [1.1](#)
 - [1.2](#)

Text .

Required ☒

Type of question Choice

answer

Number of attempt

Range step

Range min

Range max

Answer 1

[Add other answer](#)

SubQuestion

Options game

Type of question Part two

It is with money real

feedback

[Add Question](#)

None

Part two

Decision One v1

Decision One v2

Decision Two

Decision Three

Decision Four

Decision Five

Decision Six

Figure B.21: Preguntas para los distintos juegos

B. MANUAL DE INVESTIGADOR

- Decision one v1: Una pregunta que implementa una versión de un juego de lotería.
- Decision one v2: Una pregunta que implementa una segunda versión de un juego de lotería.
- Decision two: Una pregunta que implementa un juego de renta.
- Decision three: Una pregunta que implementa otro juego de renta.
- Decisión four: Una pregunta que implementa el juego del ultimatum, decidiendo el jugador la cantidad de dinero enviada al otro jugador.
- Decision five: Una pregunta que implementa el juego del ultimatum, decidiendo el jugador si acepta la cantidad enviada por otro usuario.
- Decision six: Una pregunta que implementa el juego del dictador.

Todos os juegos anteriores se pueden jugar con dinero real y dinero no real.

Además se le puede ofrecer al usuario la posibilidad de recibir feedback sobre las decisiones que ha tomado en los juegos.

B.2.5. Editar y eliminar preguntas

Después de crear una pregunta existe la posibilidad de editar y eliminar preguntas. Tenga en cuenta que si eliminas una pregunta, todas las preguntas que dependen de está serán eliminadas. Aun así la plataforma le mostrará un aviso de las preguntas que se eliminarán. Lo mismo sucede si modificas el tipo de pregunta de la que dependen otras.

Parte1 / 1.1 /

Add Question!!

- Parte1
 - 1.1
 - 1.2

Text

```
## This is a example of Markdown  
**Markdown** is rendered on the fly in the  
<i>preview area</i>!
```

More about [markdown](http://daringfireball.net

This is a example of Markdown

Markdown is rendered on the fly in the *preview area*!

More about [markdown](#).

Required ☒

Type of question

YES/NO

answer

none if There isnt correct answer

Number of attempt

SubQuestion

Options game

Add Question

Pregunta 1

[Edit](#) [Delete](#)

Pregunta 2

[Edit](#) [Delete](#)

Figure B.22: Eliminar preguntas

B. MANUAL DE INVESTIGADOR

Appendix C

Guía de desarrollo

La finalidad de esta sección es dar una serie de pautas de como mejorar y añadir nuevas características a la plataforma. Se espera del lector que tenga conocimientos de las tecnologías asociadas a la plataforma, como puede ser el uso del lenguaje Python, HTML, SQLAlchemy y de la plataforma Flask. Para adentrarse en todas estas tecnologías, puede seguir el tutorial de Miguel Grinberg <http://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>

C.1. Preguntas de tipo fecha

En esta sección vamos a tratar de como incluir una pregunta que se espera una respuesta de tipo fecha.

C.1.1. Modificación de la base de datos

En el fichero *models.py* podemos encontrar el mapeo objeto-relacional, ORM, de la aplicación. En el podemos encontrar la clase *Question*, con todos los métodos y atributos asociadas a las preguntas, además podemos encontrar una serie de clases, que heredan de *Question* e implementa los métodos y atributos asociados a un tipo de preguntas. A nivel de tablas, se ha elegido que tanto la clase *Question* como todas las clases que heredan de *Question* se hallen la misma tabla.

Para crear un nuevo tipo de pregunta bastaría con la siguiente declaración:

```
1 class QuestionDate(Question):  
2     '''Question of type date  
3     '''  
4     __mapper_args__ = {'polymorphic_identity': 'date'}
```

Además deberíamos de crear los métodos para importar y exportar este tipo de preguntas a XML. En este caso, como no tiene ningún atributo especial no haría falta nada.

C. GUÍA DE DESARROLLO

Por otra parte, para guardar la respuesta a este tipo de preguntas, tenemos varias opciones, desde crear una subclase de tipo *AnswerDate* que herede de *Answer* o simplemente añadir un campo de tipo *Date* a la clase *Answer*. Esta última opción ha sido la elegida:

```
1 class Answer(db.Model):
2     '''A table with answers'''
3     __tablename__ = 'answer'
4     ...
5     answerDate = Column(Date)
```

Para añadir los cambios en la base de datos, debemos de migrar la base de datos y actualizarla, con los siguientes comandos:

```
1 (venv)$ ./manage.py db migrate
2 (venv)$ ./manage.py db upgrade
```

C.1.2. Modificación del módulo Researcher

Para que el investigador pueda añadir de manera visual este tipo de preguntas en su encuesta, tendremos que modificar los siguientes ficheros:

- *app/researcher/views.py*: En este fichero se encuentra la declaración de todas las funciones que puede hacer uso el investigador, normalmente cada función se representa en una página web.
- *app/researcher/forms.py*: En este fichero se encuentra la declaración de todas las clases de formularios que puede hacer uso el módulo researcher.
- *app/templates/researcher/add_edit_question.html*: definición de la plantilla que se usa para mostrar las opciones de añadir/editar preguntas.

Empezaremos por añadir el tipo de pregunta a la lista de preguntas disponibles, en el fichero *forms.py*:

```
1 class QuestionForm(Form):
2     listQuestionType = [('yn', 'YES/NO'),
3                         ('text', 'Text'), ('choice', 'Choice'), ('likertScale', 'Likert Scale'), ('date', 'Date')]
```

En el fichero *views.py*, procedemos a modificar la definición de *select_type*, que se encarga de leer el formulario de añadir/editar preguntas y generar un objeto del tipo *Question* deseado:

```
1 def selectType(form, section):
2     if form.questionType.data == 'date':
3         question = QuestionDate()
```

En el fichero *add_edit_question.html*, esta la plantilla usada para mostrar las opciones disponibles. Además del código HTML, hay un poco de código JavaScript, para ayudar al investigador a la hora de rellenar las opciones, ocultando todas aquellas opciones (formularios) que no son necesarios para el tipo de pregunta.

En nuestro caso, interesa ocultar todas las opciones, cada vez que se selecciona una pregunta de tipo fecha, para ello añadiremos el siguiente código a la función *onchange_question*:

```

1 function onchange_question(selValue)
2 {
3     $("#divText").hide();
4     $("#divLikert").hide();
5     $("#divAnswer1").hide();
6     $("#divAnswer2").hide();
7     $("#divAnswers").hide();
8     $("#divRange").hide();
9     $("#divRender").hide();
10    $("#divContainer").hide();
11    $("#divExpectedAnswer").hide();
12    $("#divExpectedAnswer").hide();
13    $("#collapseSubquestion").collapse("hide");
14    if (selValue=="date")
15    {
16        //nothing to do, all options hide by default
17    }

```

Con esto hemos terminado las modificaciones del módulo *researcher*.

C.1.3. Modificación del módulo Surveys

Deberemos de modificar los ficheros *app/surveys/forms.py* y *app/surveys.utiles.py*.

Lo primero que debemos de decidir, es como representar la fecha al usuario, en este caso hemos elegido el formulario de fecha por defecto que trae *WTForms*, *wtforms.fields.DateField*, que comprobará automáticamente si la fecha es válida.

Si se desea implementar un campo y validador no disponible por *WTForms*, se pueden crear las clases necesarias, para ello consulte <http://wtforms.readthedocs.org/en/latest/> o vea los formularios y validadores creados para el proyecto en el fichero *app/surveys/forms.py*.

El fichero *forms.py* tiene un aspecto diferente al que hemos observado en el anterior módulo, esto es básicamente porque todos los formularios se han creado dinámicamente, debido a que es imposible saber por defecto como va a ser el formulario.

La función que nos importa es *generate_form*, básicamente deberemos decidir que hacer cuando leamos una pregunta de tipo fecha, el identificador usado es *"c"+id* de la pregunta en cuestión.

```

1     function onchange_question(selValue)
2     def generate_form(questions):
3         '''dynamically generates the forms for surveys

```

C. GUÍA DE DESARROLLO

```
4     '''
5     for question in questions:
6         if isinstance (question, QuestionDate):
7             if question.required:
8                 setattr(AnswerForm, "c"+str(question.id), DateField('Answer', validators
9                                     = [Required()])))
10            else:
                setattr(AnswerForm, "c"+str(question.id), DateField('Answer', validators
                                    = [Optional()])))
```

En el fichero útiles deberemos de modificar la función *new_answer()* el cual genera un objeto de tipo *Answer* dependiendo del tipo de pregunta:

```
1 def new_answer(question, form, user):
2     if isinstance (question, QuestionDate):
3         answer = Answer (answerDate = form["c"+str(question.id)].data, user= user,
4                             question = question)
5         answer.answerText = str(answer.answerDate)
```

En el atributo `answer.answerText` siempre guardamos una representación del texto de la respuesta. Esto también se puede implementar con un atributo híbrido en el ORM.

Con esto ya hemos terminado una implementación básica de una pregunta de tipo fecha.

C.2. Preguntas de tipo casillas de verificación

En esta sección vamos a tratar brevemente de como incluir una pregunta, en la que puede haber múltiples respuestas.

C.2.1. Modificación de la base de datos

Lo primero que debemos de decidir es como almacenar todas las respuestas posibles a la pregunta, ya sea creando una clase nueva con todas las respuestas posibles a una pregunta, o almacenar en la pregunta todas las respuestas posibles, ya sea usando una lista (*PickleType*), *JSON*, o cualquier formato que se nos ocurra.

En este caso hemos decidido implementarlo en una tabla nueva, para ello creamos primero la clase nueva, a la cual hemos añadido un nuevo atributo con el mínimo numero de opciones a marcar en la respuesta.

```
1 class QuestionMultipleChoices(Question):
2     '''Question with multiple choices
3     '''
4     __mapper_args__ = {'polymorphic_identity': 'multipleChoices'}
5     #: Minimum number of options
6     min_choices = Column(Integer, default = 0)
7     choices = relationship('Choice',
8                             cascade="all, delete-orphan",
9                             backref = 'question', lazy = 'dynamic')
```

C.2 Preguntas de tipo casillas de verificación

Después creamos una nueva clase con todas las opciones posibles para una pregunta:

```
1 class Choice (db.Model):
2     '''A table with Choices to a Question with multiple Choices
3     '''
4     __tablename__ = 'choice'
5     #: unique id (automatically generated)
6     id = Column(Integer, primary_key = True)
7     #: Text for this choice
8     text = Column(String, nullable = False)
9     question_id = Column(Integer, ForeignKey('question.id'))
```

Entre la clase *Answer* y la clase *Choice* tendremos una relación muchos a muchos, por lo que deberemos crear una tabla para ella, hacemos notar que creamos una tabla y no una clase nueva, ya que solo lo usaremos para crear una relación entre *Answer* y *Choice*, sin ningún atributo nuevo.

```
1 class Answer(db.Model):
2     '''A table with answers
3     '''
4     __tablename__ = 'answer'
5     ...
6     choices = relationship("Choice",
7         secondary=association_answer_choices,
8         backref=backref('answers', remote_side=id),
9         lazy = 'dynamic', uselist = True)
```

Para añadir los cambios en la base de datos, debemos de migrar la base de datos y actualizarla, con los siguientes comandos:

```
1 (venv)$ ./manage.py db migrate
2 (venv)$ ./manage.py db upgrade
```

C.2.2. Modificación del módulo Researcher

Para modificar el módulo *Researcher* habrá que hacer una modificación análoga a la realizada en la modificación de añadir una pregunta de tipo fecha.

Por otro parte se podría modificar el validador del formulario para comprobar que el número de respuestas mínimas necesario es correcto, ya que no podemos pedir mas opciones de las que hay.

```
1     def validate(self):
2         if self.questionType.data == 'multiplChoices':
3             l = get_choices() #return list with choices
4             if len(l)==0:
5                 self.errors.append("There should be a choice")
6             if self.min_choices.data is not None:
7                 if self.min_choices.data>= len(get_choices()):
8                     self.min_choices.errors.append("minimum of choices must be less
9                     than the maximum")
```

Además deberemos de añadir estas opciones a la base de datos:

C. GUÍA DE DESARROLLO

```
1 def selectType(form,section):
2     if form.questionType.data == 'multiple_choices':
3         question = QuestionMultipleChoices(min_choices = form.min_choices.data)
4         for i in get_choices(): #return list with choices
5             choice = Choice(text= i, question=question)
6             db.session.add(choice)
```

C.2.3. Modificación del módulo Answer

Esta parte es análoga a la implementación de una pregunta de tipo fecha, solo que además añadiremos un validador para comprobar que el número marcado de opciones se corresponde con el mínimo exigido:

```
1 class CheckMinChoices(object):
2     '''check if the answer is the expected'''
3
4     def __init__(self, n, message=None):
5         if not message:
6             self.message = gettext("minimum of choices must be %i" %n)
7         else: # pragma: no cover
8             self.message = message
9         self.min_choices=n
10
11     def __call__(self, form, field):
12         if len(form.data)<self.min_choices:
13             raise ValidationError(self.message)
```

Además cuando creamos el formulario añadiremos el validador:

```
1 validators = [Required(), CheckMinChoices(question.min_choices)]
```

También crearemos una lista con las opciones disponibles, cuya índice será el identificador de cada opción, esta lista se la pasaremos al constructor de *wtforms.fields.SelectMultipleField*:

```
1 choices = [(str(choice.id),choice.text) for choice in question.choices]
```

Por último en el fichero útiles deberemos de modificar la función *new_answer()* el cual genera un objeto de tipo *Answer* dependiendo del tipo de pregunta:

```
1 def new_answer(question,form,user):
2     if isinstance (question,QuestionMultipleChoice):
3         answer = Answer (user= user, question = question)
4         //obtenemos la lista de elementos marcados:
5         list_aux=[]
6         for i in form["c"+str(question.id)].data:
7             //obtenmos la opción marcada
8             choice = Choice.query.get(i)
9             //añadimos la opción a la respuesta
10            answer.choices.append(choice)
11            //guardamos en una lista auxiliar la respuesta para guardarla también
            //como texto
12            //pero no sería necesario
13            list_aux.append(choice.text)
14            answer.text = ', '.join(list_aux)
```


C.3. Encuestas en varios lenguajes

Puede ser muy útil tener una encuesta en múltiples idiomas, para ello la solución mas fácil y de las mas cómodas para el investigador es la de crear la encuesta en un idioma predeterminado y luego traducir esta a otros idiomas. La traducción se podría hacer con un simple editor de texto o integrando esta en la aplicación, para lo que bastaría crear una página con todos los textos de una encuesta e ir traduciendo.

La forma mas sencilla sería almacenar en un formato como JSON o XML el texto y el idioma al que pertenece. Todo ello guardado en el mismo atributo. Esta podría ser una representación de un texto guardado:

```
1 {  
2     {"language": "eng",  
3      "text": "Hello wolrd"},  
4     {"language": "es",  
5      "text": "Hola Mundo"},  
6 }
```

Para no cambiar todos los accesos a los textos, podemos hacer uso de *@hybrid_property*, para devolver solo el texto en el idioma deseado o en su defecto en el predeterminado. La plataforma ya hace uso de *Babel*, que es una colección de herramientas que sirve para la internacionalización de las aplicaciones escritas en Python, entre las funciones que tiene hay una que nos devuelve el idioma deseado por el usuario/navegador:

```
1 request.accept_languages.best_match(LANGUAGES.keys())
```

Por lo que una clase como Survey quedaría de la siguiente manera:

```
1 class Survey(db.Model):  
2     ....  
3     text_json = Column(String, nullable = False)  
4     @hybrid_property  
5     def survey(self, language = request.accept_languages.best_match(LANGUAGES.keys()  
6         )):  
7         return get_text(self.text_json, language)
```

C.4. Encuestas con preguntas que saltan a distintas secciones

Otra mejora interesante para la plataforma, es saltar a alguna sección en concreto dependiendo de la respuesta dada a una pregunta.

C.4.1. Modificación de la base de datos:

Para ello primero deberemos de crear un identificador único para cada sección especial. Podemos pensar en usar el identificador de cada sección, pero esto no es buena idea, ya que es la base de datos quien se encarga de administrarlo y puede ser poco amigable, por lo que decidimos que el nombre de la sección sea único.

```
1 class Section(db.Model):
2     ...
3     title = Column(String(128), nullable = False)
4     #:nos aseguramos que el titulo de la sección sea único en la encuesta
5     __table_args__ = (UniqueConstraint('title', 'survey_id'),)
```

Además modificaremos las preguntas de tipo test, *QuestionChoice*, para que dependiendo de la respuesta dada saltemos a una sección u otra, para facilitar el diseño usaremos la implementación realizada para las preguntas con casillas de verificación, en vez de guardar los posibles resultados en una lista.

```
1 class QuestionChoice(Question):
2     ...
3     choices = relationship('Choice',
4                             cascade="all, delete-orphan",
5                             backref = 'question', lazy = 'dynamic')
6
7 class Choice (db.Model):
8     '''A table with Choices to a Question with multiple Choices
9     '''
10    __tablename__ = 'choice'
11    #: unique id (automatically generated)
12    id = Column(Integer, primary_key = True)
13    #: Text for this choice
14    text = Column(String, nullable = False)
15    question_id = Column(Integer, ForeignKey('question.id'))
16    #: id de la sección a la cual se realizará el salto.
17    section_id = Column(Integer, ForeignKey('section.id'))
18    ## Relationships
19    section = relationship("Section")
20    answer = relationship("Answer")
```

Por último modificamos la clase *Answer*, para que pueda tener una relación con la clase *Choice*

```
1 class Answer (db.Model):
2     choice = Column(Integer, ForeignKey('choice.id'))
```

C.4.2. Modificación del módulo Researcher:

Deberemos de modificar los formularios de añadir/editar sección, añadiendo la posibilidad de definir un identificador, por otra parte también se modificara la plantilla, */templates/resarcher/addEditSection.html* y la vista.

Por otra parte también habrá que modificar las opciones dadas en las preguntas de tipo test, permitiendo asociar a cada respuesta un salto a una sección dada. Para ello podemos usar el campo definido por la clase *wtforms.ext.sqlalchemy.fields.QuerySelectField*

C.4 Encuestas con preguntas que saltan a distintas secciones

```
1 section = QuerySelectField ('Section',get_label='text',validators=[Optional()])
```

y para rellenar el formulario, que estará formado por la tupla *Section.id* y *title*, título de la sección:

```
1 form.section.query=Question.query.filter(  
2     Survey.id == id_survey,  
3     Section.root == Survey,  
4     Section.title!=id_section //evitamos poder saltar a la misma sección  
5 )
```

C.4.3. Modificación de la clase StateSurvey

Por último habrá que cambiar algo de la lógica de control de la encuesta, para poder realizar saltos a la sección indicada dependiendo de la respuesta dada. Cada usuario cuando empieza una encuesta, se le asocia una clase a esa encuesta *StateSurvey*, que guarda la información relativa del usuario a esta encuesta, como puede ser si ha terminado la encuesta, cuanto tiempo lleva empleada en ella o por que sección va.

La clase tiene un método llamado *finishedSection*, el cual se llama cuando se termina una sección y decide cual es la siguiente sección a realizar. Por lo que este sería un buen lugar para comprobar las respuestas dadas en la sección para comprobar cual es la siguiente sección a realizar.

Primero buscamos la respuesta dada a la pregunta de la cual dependerá la próxima sección, para simplificar el proceso hemos supuesto que solo existe una pregunta de este tipo por sección y que será obligatoria.

```
1 answer = Answer.query.filter(  
2     //obtenmos las preguntas de tipo Choice de la sección en la que estamos  
3     QuestionChoice.section_id==self.get_section,  
4     //obtenmos las posibles respuestas a la pregunta de tipo choice  
5     Choice.question==QuestionChoice,  
6     //Las posibles elecciones deben tener asociada un salto  
7     Choice.section!=None,  
8     // buscamos todas las respuestas del usuario  
9     Answer.user_id==Self.user_id,  
10    //y cogemos solo la que coincide con la pregunta  
11    Answer.question==QuestionChoice  
12 ).first()
```

Por lo que en *answer* tenemos la respuesta dada por el usuario de la que dependerá la siguiente sección.

```
1 self.nextSection == answer.choice.section
```

C.5. Plataforma en la nube

Aquí vamos a tratar de como desplegar nuestra plataforma en la nube. Hemos elegido *Heroku*, no sólo porque es uno de los mas populares de la red, sino porque además ofrece también un nivel de servicio gratuito.

Para implementar la plataforma web en *Heroku*, es tan fácil como subir la aplicación usando Git. Para las aplicaciones desarrolladas en Python se espera un fichero *requirements.txt* en la que se encuentran todos los módulos que deben instalarse.

Primero nos crearemos una cuenta en Heroku y nos bajaremos el "*cliente Heroku*", una vez instalado el cliente, accederemos a nuestra cuenta y clonaremos nuestra plataforma desde github y crearemos la plataforma en Heroku:

```
1 $ heroku login
2 $ git clone git://github.com/nu_kru/swarm-survey.git
3 $ cd swarm-survey
4 $ heroku create swarm-survey Creating swarm-survey... done, stack is cedar http://
  swarm-survey.herokuapp.com/ | git@heroku.com:swarm-survey.git
```

Con esto ya tendríamos nuestra plataforma en la nube. Pero aun no hemos terminado, ya que Heroku impone unas cuantas restricciones:

- Las aplicaciones que se ejecutan en Heroku no escriben archivos permanentemente en el disco.
- La base de datos a usar debe ser PostgreSQL
- No proporciona un servidor web

C.5.1. Migración del log

Debido a que las aplicaciones que se ejecutan en Heroku no pueden escribir permanentemente en el disco, el log de la plataforma lo perderíamos cada cierto tiempo, para ello lo solucionaremos usando el log que nos proporciona Heroku, es mas si se observa el fichero *config.py*, ya tenemos escrita una configuración por si se lanza la plataforma en Heroku, que se encarga de modificar el log a los requisitos de la nube

```
1 //log to stderr
2 import logging
3 from logging import StreamHandler
4 file_handler = StreamHandler()
5 file_handler.setLevel(logging.WARNING)
6 app.logger.addHandler(file_handler)
```

Para ver el log simplemente desde el cliente escribimos:

```
1 $ heroku logs
```

Esto nos mostrara todos los logs, tanto de la nube como de nuestra aplicación, si solo queremos ver la de la aplicación:

```
1 $ heroku logs --source app
```

C.5.2. Migración a PostgreSQL

La plataforma de Heroku nos proporciona una URI por la cual podemos acceder a nuestra base de datos, como durante el desarrollo hemos usado SQLAlchemy, y no hemos usado características únicas solo disponible en ciertas bases de datos, la migración a PostgreSQL es transparente, sin tener que realizar ninguna modificación en el código.

Por otra parte si se observa el fichero de configuración, la base de datos la obtenemos de una variable de entorno si está definida o sino de la ruta indicada por nosotros. Esta variable de entorno es la misma que usa Heroku para darnos la dirección de la bbdd por lo que no tendremos que realizar ningún cambio.

```
1 class ProductionConfig(Config):
2     SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL') or \
3         'sqlite:/// ' + os.path.join(basedir, 'data.sqlite')
```

C.5.3. Servidor web

Ya que Heroku no nos proporciona ningún servidor web, en su lugar espera que la aplicación ejecute su propio servidor en el puerto indicado por la variable de entorno *\$PORT*

El servidor web proporcionado por Flask para el desarrollo no es conveniente usarlo en producción, ya que no es multihilo, en la documentación proporcionada por Heroku en las aplicaciones Python sugieren la instalación de *gunicorn*, que es un servidor web escrito en Python, para hacer uso de él, lo podemos instalar vía *pip*, para ejecutar el servidor con la aplicación tan solo debemos escribir:

```
1 gunicorn manage.py runserver:app
```

Con esto ya tendríamos todo lo necesario para ejecutar nuestra aplicación en Heroku. Si se observa detenidamente las modificaciones a hacer en la plataforma son nulas, sobre todo gracias a la buena definición del fichero config.py, en el que hemos tenido en cuenta las distintas configuraciones de la aplicación, pasando por el modo de desarrollo al modo producción, con tres variantes posibles dependiendo de si la máquina contiene un sistema tipo UNIX, está en la nube o ninguno de estos casos.

C.6. Consideraciones en el desarrollo de la plataforma

Durante el desarrollo de aplicaciones en Python se recomienda instalar todos los módulos a usar en entornos virtuales, además en nuestro caso como hemos hecho uso de pip para la instalación de cada módulo, podemos obtener una lista de todos los módulos instalados usando:

```
1 (venv)$ pip freeze > requirements.txt
```

Appendix D

Análisis de requisitos de la aplicación

En este anexo se presenta una descripción del análisis de requisitos que se llevo a cabo antes de realizar la implementación de la plataforma y durante las revisiones de esta. A lo largo del anexo se mostrará el documento de especificaciones de requisitos. Este documento no ha sido creado con el fin de ser completamente riguroso a las convenciones, aunque más de una vez se sigan, sino que el fin último es que el lector comprenda este documento.

D.1. Especificación de Requisitos Software

En este apartado vamos a hablar de los requisitos software que debe cumplir nuestra plataforma. Estos requisitos se elaboraron después del estudio de las herramientas actuales y el documento "*¿Cómo son nuestros voluntarios?*", por otra parte se ha intentado seguir las recomendaciones dadas en la Especificación de Requisitos según el estándar IEEE 830

D.1.1. Introducción

D.1.1.1. Propósito

El propósito es definir cuáles son los requerimientos que debe tener la plataforma para elaboración de encuestas, así como la adaptación de está al experimento "*¿Cómo son nuestros voluntarios?*"

La aplicación se desarrollara como proyecto final de carrera y permitirá el desarrollo del experimento anteriormente citado, así como ayudar a la comunidad científica en la elaboración de encuestas mas o menos complejas, en los que se quiera aleatorizar el orden de las secciones, así como la existencia de secciones excluyentes.

D. ANÁLISIS DE REQUISITOS DE LA APLICACIÓN

D.1.1.2. Ambito

La plataforma debe permitir la creación y la manipulación de encuestas complejas, así como la posibilidad de limitar el tiempo y el numero de usuarios a la hora de realizar las encuestas.

Una vez definida y publicadas estas encuestas, se permitirá a los usuarios encuestados la realización de está, registrando en todo momento los tiempos tomados para la resolución de cada pregunta.

En el caso de del experimento "*¿Cómo son nuestros voluntarios?*", se le permitirá al usuario recibir feedback respecto a la toma de sus decisiones, con respecto a las respuestas de los demás usuarios.

Además en dicho experimento, en casos elegidos de forma aleatoria, el voluntario recibirá un pago asociado a una de sus decisiones.

Los usuarios de la plataforma no deberán de poseer algún conocimiento específico para el uso de esta.

Por otra parte la plataforma deberá de estar bien documentada para facilitar la ampliación de posibilidades de esta.

D.1.1.3. Definiciones, Acrónimos y Abreviaturas

OpenID: estándar de identificación digital descentralizado, con el que un usuario puede identificarse en una página web a través de una URL (o un XRI en la versión actual) y puede ser verificado por cualquier servidor que soporte el protocolo.

Lenguaje de marcas: forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

XML: siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas

CSV: siglas del inglés comma-separated values, son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas u otro identificador predefinido y las filas por saltos de línea. Los campos que contengan una coma, un salto de línea o una comilla doble deben ser encerrados entre comillas dobles.

Lenguaje de marcas ligero: tipo de formateo de texto más o menos estandarizado, que ocupa poco espacio y es fácil de editar con un editor de texto.

Markdown: lenguaje de marcado ligero que trata de conseguir la máxima legibilidad tanto en sus forma de entrada como de salida.

D.1.2. Descripción general

D.1.2.1. Perspectiva del Producto

La plataforma a desarrollar no estará relacionado con otros productos, aunque se adaptará esta al desarrollo del experimento "¿Cómo son nuestros voluntarios?"

D.1.2.2. Funciones del Producto

Las funciones que debe realizar la plataforma a grandes rasgos son:

- Creación y edición de encuestas complejas.
- Registro de todas las respuestas así como sus tiempos.
- Exportar los resultados obtenidos en las encuestas.
- Ofrecer feedback a los usuarios

D.1.2.3. Características de los usuarios

Existen tres tipos de usuarios:

- Los investigadores para la elaboración de encuestas complejas, poniendo especial hincapié en la aleatorización y diversificación de las secciones de una encuesta. Solo deberán tener conocimiento alguno en el manejo de un navegador web, por otra parte se ofrecerá un manual con todas las opciones de la plataforma.
- Voluntarios que responden a los cuestionarios, que solo deberán tener conocimiento alguno del manejo de un navegador web.
- Desarrolladores que desean adaptar y ampliar la plataforma a sus necesidades, para ello a parte de este documento, se incluye un manual de desarrollo. Deberán estar familiarizados con las tecnologías usadas.

D.1.2.4. Restricciones

La aplicación se debe poder acceder desde un portal web. Así mismo los investigadores nunca obtendrán información identificativa de los voluntarios que participen en su plataforma. Deberá funcionar la plataforma bajo una máquina gobernada por Linux

D. ANÁLISIS DE REQUISITOS DE LA APLICACIÓN

D.1.2.5. Requisitos futuros

Debe pensarse que la plataforma se debe poder adaptar a otras cuestionarios con otros requisitos, por lo que debe estar bien documentada y tener en cuenta estas posibles necesidades. Como podría ser la creación de algún tipo de pregunta no contemplada en la plataforma actual.

También se podría considerar la elaboración de un lenguaje específico del dominio para la creación de encuestas.

D.1.3. Requerimientos específicos

D.1.3.1. Interfaces Externas

Se deberá de acceder a la plataforma mediante un navegador web. Esta deberá de adaptarse al tamaño del dispositivo por el cual se accede a la plataforma, nunca siendo inferior a una resolución de 480px para su correcta visualización. Por otra parte al ser una plataforma web, se deberá tener acceso a través de la red al servidor donde esté alojado la plataforma.

La plataforma usará un sistema de direcciones web amigables, para facilitar el manejo de esta a través del navegador web.

D.1.3.2. Requerimientos funcionales

Agruparemos los requisitos funcionales de la plataforma según el usuario que lo esté usando, por otra parte se incluirá los requisitos que debe cumplir la adopción de la plataforma al experimento "*¿Cómo son nuestros voluntarios?*".

La plataforma permitirá a los investigadores:

- Crear nuevas encuestas.
- Añadir una fecha en la que la encuesta será visible a los voluntarios.
- Añadir una fecha limite para la realización de la encuesta.
- Añadir un número máximo de encuestas completadas.
- Añadir un tiempo máximo para contestar la encuesta.
- Añadir consentimientos a las encuestas.
- Añadir secciones a las encuestas.
- Añadir subsecciones a las secciones.

- Definir el orden de las secciones y subsecciones.
- Aleatoriar el orden de las secciones y subsecciones.
- Generar secciones y subsecciones excluyentes con la probabilidad indicada.
- Añadir preguntas a las secciones y subsecciones, estas preguntas serán del tipo:
 - Preguntas cuya respuesta es un texto.
 - Preguntas cuya respuesta es un número entero.
 - Preguntas cuya respuesta es un número decimal.
 - Preguntas cuya respuesta es Si/No.
 - Preguntas cuya respuesta es un conjunto de opciones posibles definidas por el investigador.
 - Preguntas cuya respuesta es un conjunto de opciones posibles definidas por un rango y su salto.
 - Preguntas cuya respuesta es validada por una expresión regular, indicando un mensaje de error personalizado.
 - Preguntas cuya respuesta es una escala Likert, pudiendo definir el rango de la escala, así como sendas etiquetas en los extremos de la escala Likert.
 - Preguntas de control que tienen una respuesta única, se podrá fijar un número máximo de intentos, entre uno e infinito, así como un mensaje de error personalizable.
 - Todas las preguntas anteriores descritas podrán ser obligatorias u optativas.
 - Todas las preguntas anteriores descritas podrán ser dependientes de la respuesta de otra pregunta.
 - Todas las preguntas anteriores descritas, se podrá cambiar el tipo de respuesta.
- Todas las opciones anteriormente descritas se podrán editar.
- Se usará la sintaxis Markdown para todos los campos de la encuesta.
- Exportar encuestas en formato XML.
- Importar encuestas en formato XML.

D. ANÁLISIS DE REQUISITOS DE LA APLICACIÓN

- Exportar los resultados de las encuestas a un formato CSV.

La plataforma permitirá a los voluntarios que participen en las encuestas:

- Contestar a las encuestas.
- Permitir continuar la encuesta más tarde.
- En caso de responder erróneamente a la pregunta, se le indicara el fallo al usuario.
- Informar del estado de la encuesta, siendo este:
 - Sin empezar.
 - Terminada.
 - Sin terminar.
 - Sin posibilidad de terminarla debido a que la fecha ha concluido.
 - Sin posibilidad de terminarla debido a que el tiempo máximo ha concluido

La plataforma permitirá a los usuarios:

- Identificarse mediante OpenID.
- Registrarse mediante correo y contraseña.
- Identificarse mediante correo y contraseña.

La plataforma registrará todos los tiempos, así como las respuestas, y numero de intentos en las preguntas de control en los cuestionarios.

En el experimento "*¿Cómo son nuestros voluntarios?*", la plataforma permitirá:

- Dar feedback a los usuarios, comparando sus respuestas con las dadas por otros usuarios.
- El voluntario podrá recibir un pago asociado a una de sus elecciones. La probabilidad de este pago viene dada por:
 - Los usuarios que realicen la parte 2 del experimento con dinero real, tendrán un 10% de recibir un pago asociado a una de sus elecciones elegidas al azar de este apartado.

D.1 Especificación de Requisitos Software

- Los usuarios que realicen la parte 3 del experimento con dinero real, tendrán un 10% de recibir un pago asociado a una de sus elecciones elegidas al azar de este apartado.
- Los usuarios que realicen la parte 2 y la parte 3 con dinero ficticio, participarán en un sorteo. Las probabilidades de ganar dinero en este sorteo viene dadas por:
 - Los usuarios tendrán un 10% de probabilidades de ganar un premio del sorteo, pudiendo ser el premio cuatro cantidades predefinidas, cada una con igual probabilidad de ser elegida.
- La plataforma avisará mediante correo a los usuarios que hayan ganado dinero.
- La plataforma se encargará de transmitir la información de los usuarios que han recibido algun tipo de compensación económica al responsable del pago.
- Los resultados de este experimento tendrán un formato CSV adaptado a las necesidades de los investigadores.

D.1.3.3. Requisitos de Rendimiento

Al tratarse de una plataforma web que permite la ejecución de múltiples usuarios al mismo tiempo, puede que existan problemas derivado a la carga de usuario. Para la realización del experimento "*¿Cómo son nuestros voluntarios?*", se espera llegar a 1000 usuarios durante un mes, por lo que en un principio no debería de existir problemas de carga, aun así se medirá en la maquina de producción la carga máxima de usuarios que soporta el sistema y el tiempo de respuesta.

D.1.3.4. Atributos del Sistema

Se espera una alta fiabilidad del sistema, aun así se registrará cualquier incidente en la plataforma pudiendo avisar al desarrollador y administrador de la plataforma.

También se espera en un futuro la ampliación de la plataforma así como el uso de esta en otras investigaciones, por lo que debe estar bien documentada para facilitar su adopción.

En cuanto a la seguridad, como ya hemos comentado anteriormente se registrará cualquier incidente y acceso a la plataforma. En el sistema existirán tres tipos de usuarios, el administrador, que deberá tener acceso al servidor donde esté alojado

D. ANÁLISIS DE REQUISITOS DE LA APLICACIÓN

la plataforma. Y los investigadores y usuarios que respondan a las encuestas, que podrán entrar en el sistema mediante OpenID, o mediante un login y contraseña.

Appendix E

Diseño de la aplicación

En este anexo se presenta una descripción del diseño que se llevo a cabo antes de realizar la implementación de la plataforma y durante las revisiones de esta. A lo largo del anexo se mostrará el modelado de la base de datos, diagrama de clases, casos de usos y traza de eventos, así como la descripción de los distintos módulos creados. Este documento no ha sido creado con el fin de ser completamente riguroso a las convenciones, aunque más de una vez se sigan, sino que el fin último es que el lector comprenda este documento.

E.1. Diagrama de Clases

En un primer lugar se va a mostrar sin profundizar el diagrama de clases que hace uso de la base de datos, representando únicamente el modelo estático del sistema, el cual se puede ver en la figura E.1

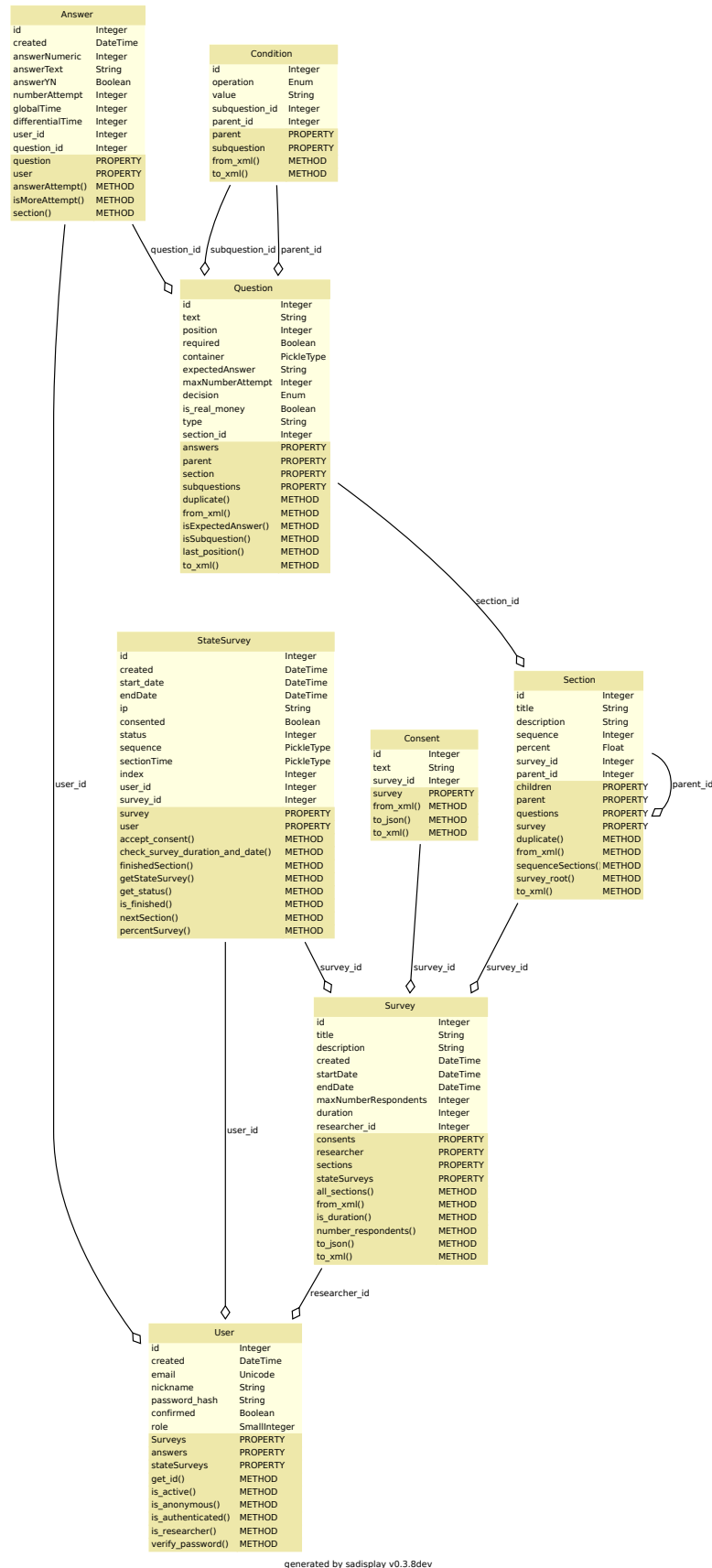
Observando con mas detenimiento la clase *Question*, cada tipo de pregunta desciende de la clase *Question*, en la figura E.2 se puede ver con mas detalle.

Por otra parte, para el experimento "*¿Como son nuestros voluntarios?*" Se creo una serie de clases para los distintos juegos y sorteos, como se puede ver en la figura E.3.

En la figura E.4 se puede ver con mas detenimiento la clase *Game*.

Como hemos dicho anteriormente estos diagramas solo hacen referencia al modelo estático del sistema, no así a las demás clases que interactúan con estas, como puede ser los distintos formularios y validadores web usados para la representación del modelo.

E. DISEÑO DE LA APLICACIÓN



E.1 Diagrama de Clases

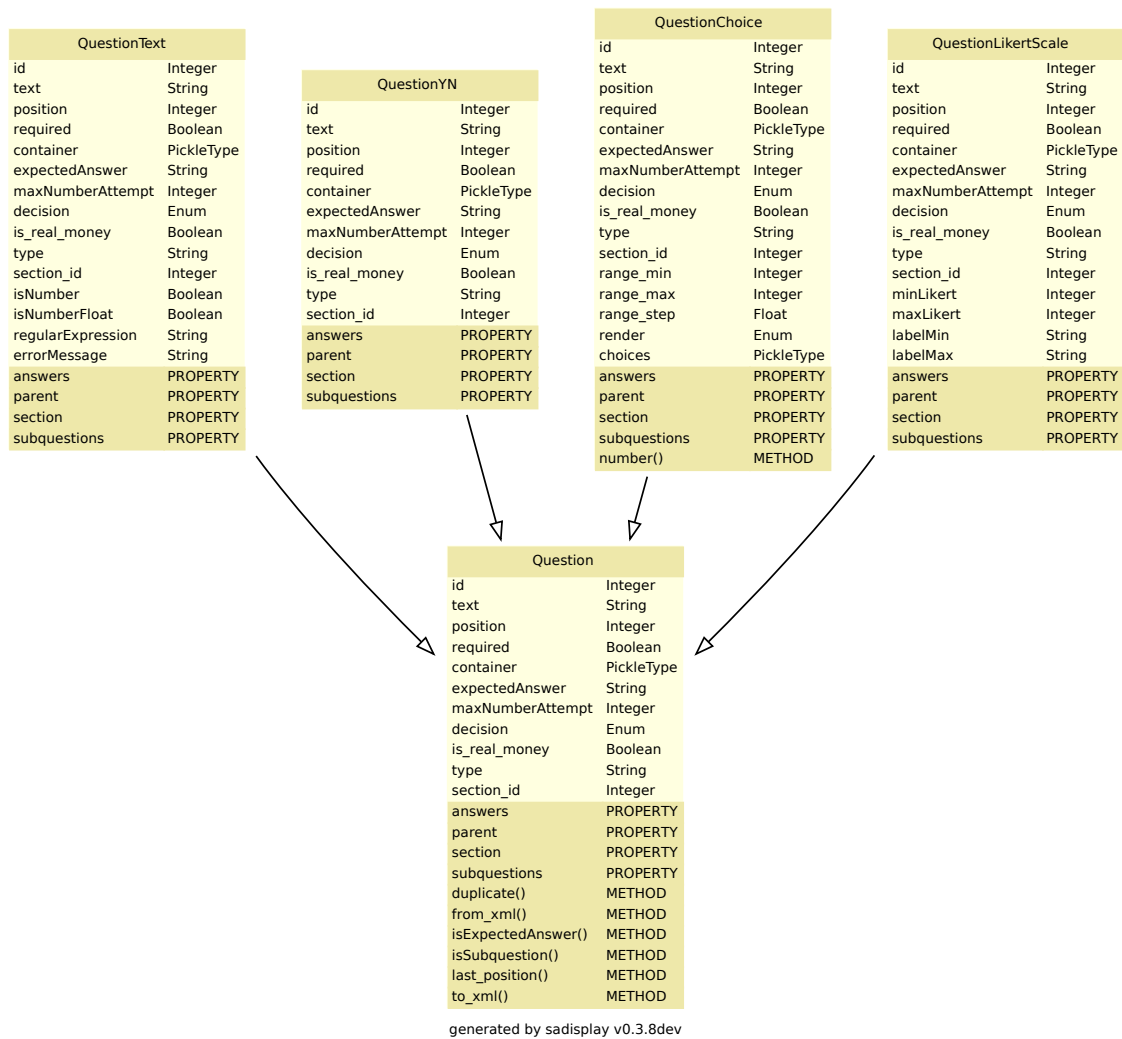
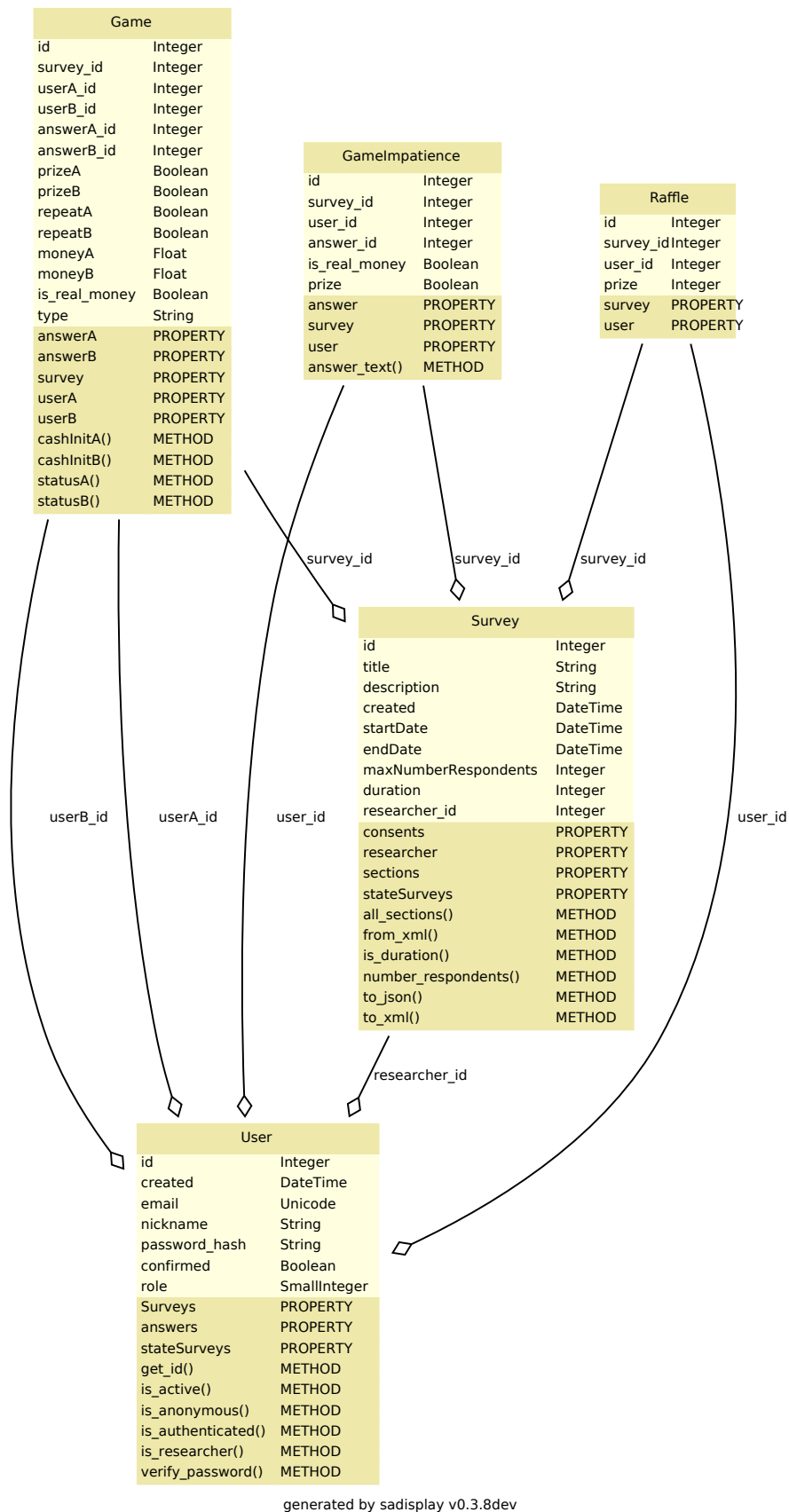


Figure E.2: Diagrama de clases de Question

E. DISEÑO DE LA APLICACIÓN



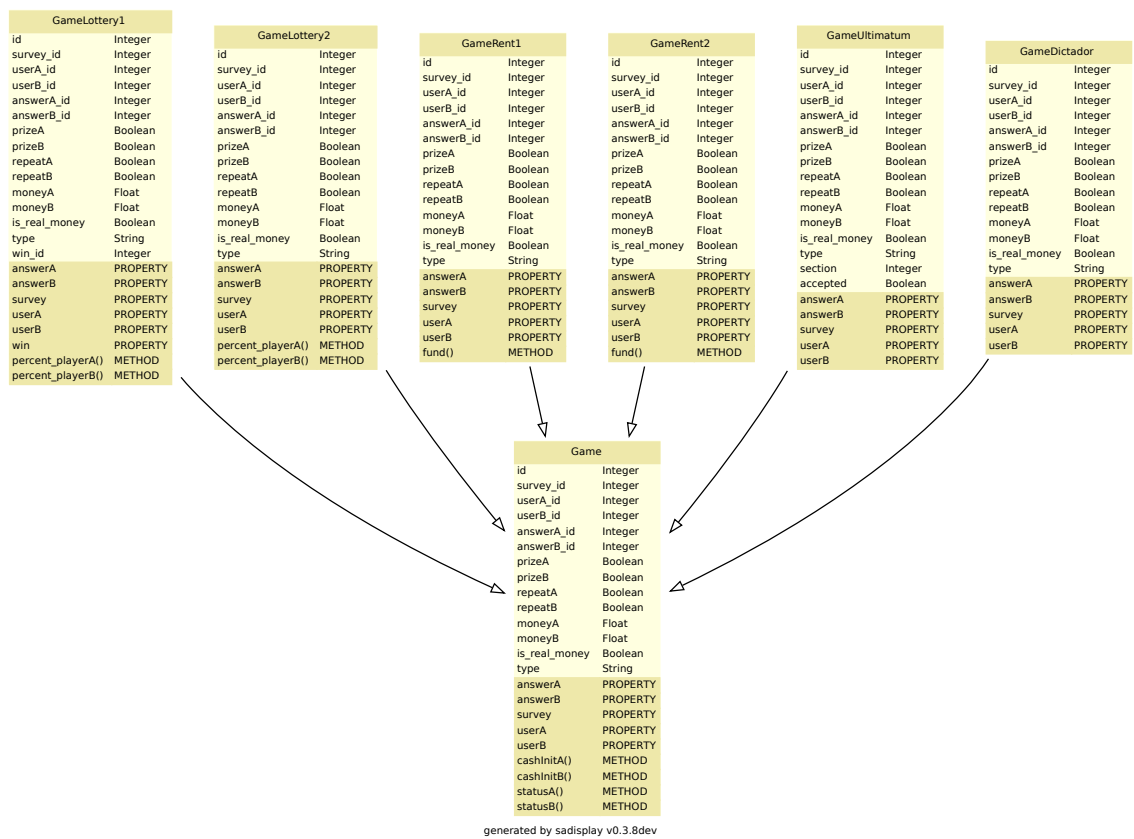


Figure E.4: Diagrama detallado de la clase de Game

E. DISEÑO DE LA APLICACIÓN

E.1.1. Entidades del sistema

A continuación vamos a describir las entidades del sistema:

E.1.1.1. User, usuario

Un usuario es una persona que se registra en la plataforma para poder hacer uso de ella, ya sea creando encuestas o contestándolas. Además guarda los permisos que posee dicho usuario dentro de la plataforma, pudiendo tener un rol de investigador o de voluntario.

E.1.1.2. Survey, encuesta

Survey sería la cabecera de la encuesta, en la que están definidas los atributos y las limitaciones de ésta, como puede ser la fecha durante la que estará activa la encuesta, la duración de esta, o simplemente el título y la descripción de esta.

E.1.1.3. Consent, consentimiento

Consent almacena los consentimientos que debe aceptar un usuario para realizar una encuesta. Cada consentimiento pertenece a una encuesta.

E.1.1.4. Section, sección

Section sirven para definir las distintas secciones organizativas que posee una encuesta, a su vez una sección puede contener más secciones. Además en cada sección se puede definir si son secciones exclusivas respecto a las secciones del mismo nivel, así como definir el nombre y la probabilidad de realizar dicha sección de la encuesta. Al final las secciones generan un árbol que cuelga de una encuesta.

E.1.1.5. Question, pregunta

Question almacena una pregunta de la encuesta. Las preguntas pueden ser de distinto tipo, cada una tiene su propia clase que hereda de *Question*. Cada pregunta está definida dentro de una sección.

QuestionText, preguntas de texto Almacena preguntas cuya respuesta es texto, almacena también las limitaciones que pueda tener la respuesta, como puede ser si es un número entero, decimal o viene limitado por una expresión regular.

QuestionYN, preguntas de si o no Hace referencia a las preguntas cuya respuesta es si o no.

QuestionChoice, preguntas de elección Contiene las distintas opciones posibles en las preguntas cuyo rango de respuestas está predeterminado. También guarda información de como representar estas opciones al usuario.

QuestionLikerScala, preguntas de escala likert Almacena la información necesaria para las preguntas likert.

E.1.1.6. Condition, condición

Condition hace referencia a las preguntas dependientes de la respuesta de otra pregunta. Guardando que pregunta depende de quién así como las condiciones necesarias para mostrar dicha pregunta.

E.1.1.7. Answer, respuesta.

Guarda la respuesta que ha dado un usuario a una pregunta, así como el tiempo empleada en ella.

E.1.1.8. StateSurvey, estado de la encuesta

Almacena la información relativa del estado en el que se encuentra una encuesta por parte de un usuario, guardando entre otras cosas la sección por la que va, el tiempo empleado o el estado de la encuesta.

E.1.2. Entidades del sistema del proyecto "¿Cómo son nuestros voluntarios?"

A continuación vamos a describir las entidades creadas para el proyecto "*¿Cómo son nuestros voluntarios?*"

E.1.2.1. Raffle, rifa

En el sorteo que hay dentro del proyecto, *raffe* se encarga de decidir si el usuario es premiado o no y guarda la cantidad ganada.

E. DISEÑO DE LA APLICACIÓN

E.1.2.2. **GameImpatience, juego de la impaciencia**

Hace referencia a los resultados de las decisiones tomadas en el juego de la impaciencia, que depende únicamente solo de un usuario, descrita en el anexo F.4.3

E.1.2.3. **Game**

Dentro del proyecto existen varios juegos que involucran a dos usuarios, cada juego depende de las decisiones tomadas en las respuestas de la encuesta. Cada juego hereda de esta entidad

GameLottery1, juego de la lotería 1 Guarda la información de la decisión tomada en la primera versión del juego de la lotería, descrita en el anexo F.4.4.1

GameLottery2, juego de la lotería 2 Guarda la información de la decisión tomada en la segunda versión del juego de la lotería, descrita en el anexo F.4.4.1

GameRent1, juego de la renta 1 Guarda la información de la decisión tomada en la primera versión del juego de la renta, descrita en el anexo F.4.4.2

GameRent2, juego de la renta 2 Guarda la información de la decisión tomada en la segunda versión del juego de la renta, descrita en el anexo F.4.4.3

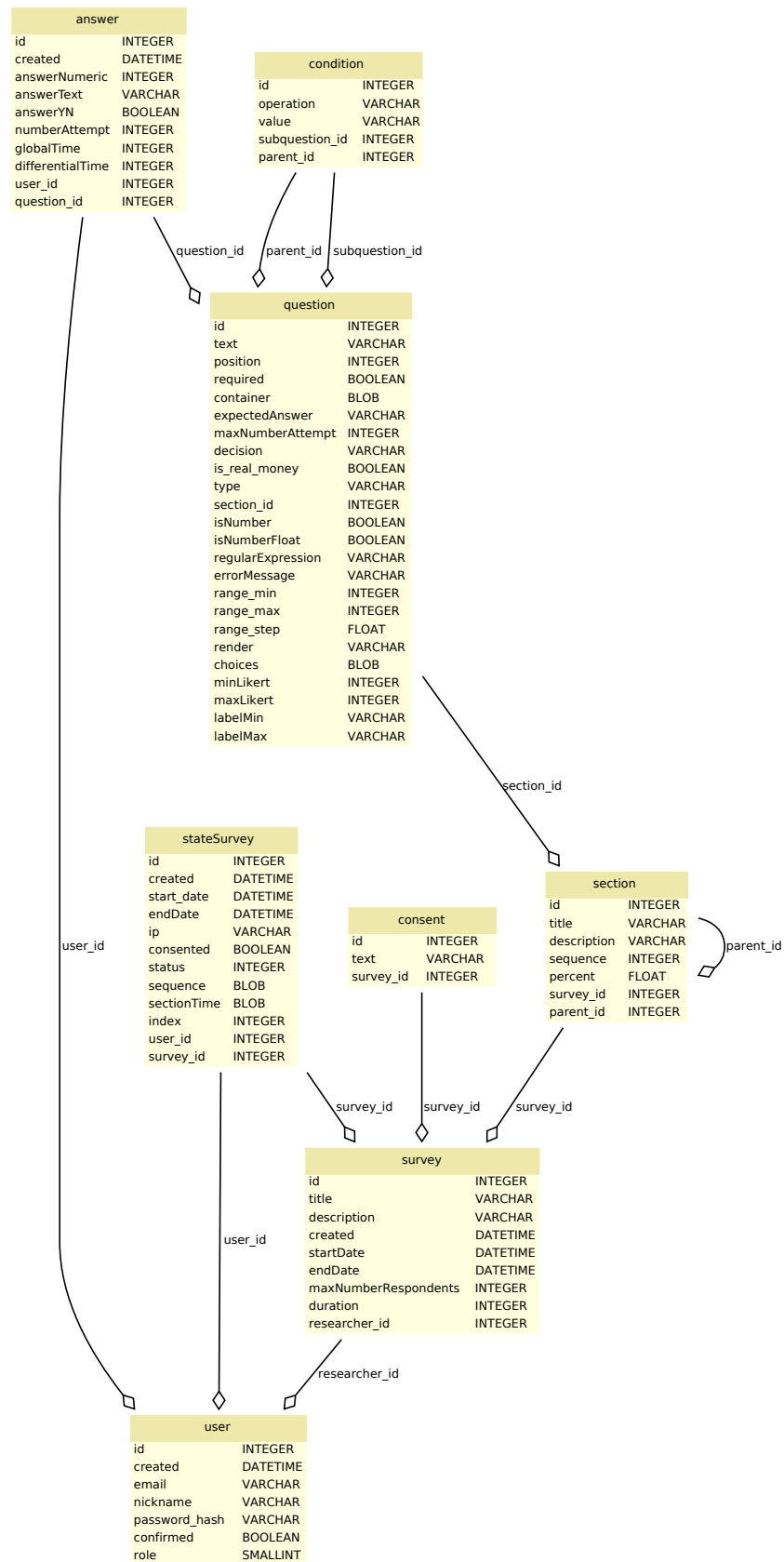
GameUltimatum, juego del ultimatum Guarda la información de la decisión tomada en el juego del ultimatum, descrita en el anexo F.4.4.4

GameDictador, juego del dictador Guarda la información de la decisión tomada en el juego del dictador, descrita en el anexo F.4.4.5

E.2. Esquema Relacional

En la figura E.5 mostramos el esquema relacional, generado por el ORM SQLAlchemy.

Durante el diseño se tuvo que tomar la decisión de como implementar las clases heredaras, si todas en la misma tabla, en distintas o algunas compartiendo tabla y otras no. Se decidió que todas compartirían la misma tabla, sobre todo a que no se espera una gran cantidad de entradas, mas comparándolo con otras tablas como puede ser *Answer*.



generated by sadisplay v0.3.8dev

Figure E.5: Esquema relacional

E. DISEÑO DE LA APLICACIÓN

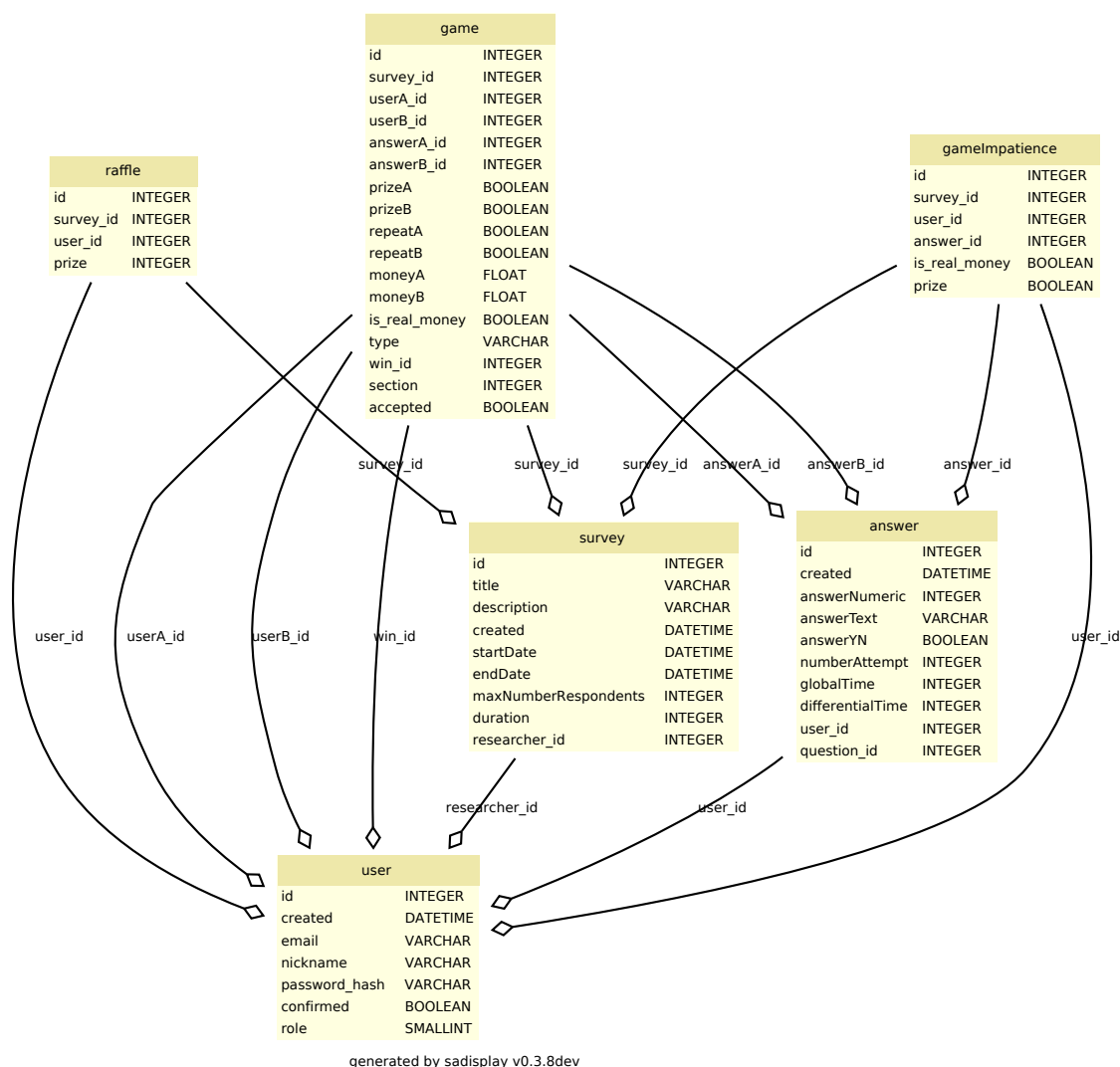


Figure E.6: Esquema relacional de los juegos

Aunque no se aprecia en el esquema, existen las siguientes restricciones en la declaración del modelo que se trasladan a la base de datos:

- Tabla **stateSurvey**, la tupla *user_id* y *survey_id* es única.
- Tabla **condition**, la tupla *subquestion_id* y *parent_id* es única.
- Tabla **answer**, la tupla *user_id* y *question_id* es única.
- Tabla **user**, el valor de *email* es único.

En la figura E.2 se muestra el esquema relacional, donde se guardan los resultados de los distintos juegos y sorteos

Las restricciones de estas tablas son las siguientes:

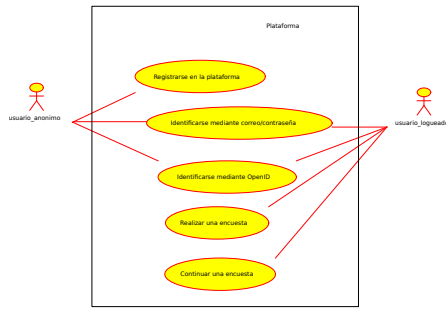


Figure E.7: Diagrama de casos de uso de un voluntario



Figure E.8: Diagrama de casos de uso de un investigador

- Tabla **gameImpatience**, la tupla *user_id* y *survey_id* es única.
- Tabla **raffle**, la tupla *user_id* y *survey_id* es única.
- Tabla **game**, la tupla *userA_id*, *userB_id* y *survey_id* es única.

E.3. Casos de Uso

En esta sección vamos a analizar los posibles usos que podrán darse a la plataforma. Para ilustrarlo vamos a emplear algunos diagramas de casos de uso y trazas de eventos.

En la figura E.3 podemos ver los posibles usos generales que le puede dar un usuario que sea un voluntario.

En la figura E.3 podemos ver los posibles usos generales que le puede dar un usuario que sea un investigador.

En la figura E.9 podemos ver los posibles usos generales que le puede dar un usuario que sea un administrador.

En la figura E.10 podemos ver una traza de eventos genérica que corresponde a la situación en la que un usuario rellena un formulario y lo envía a la plataforma,

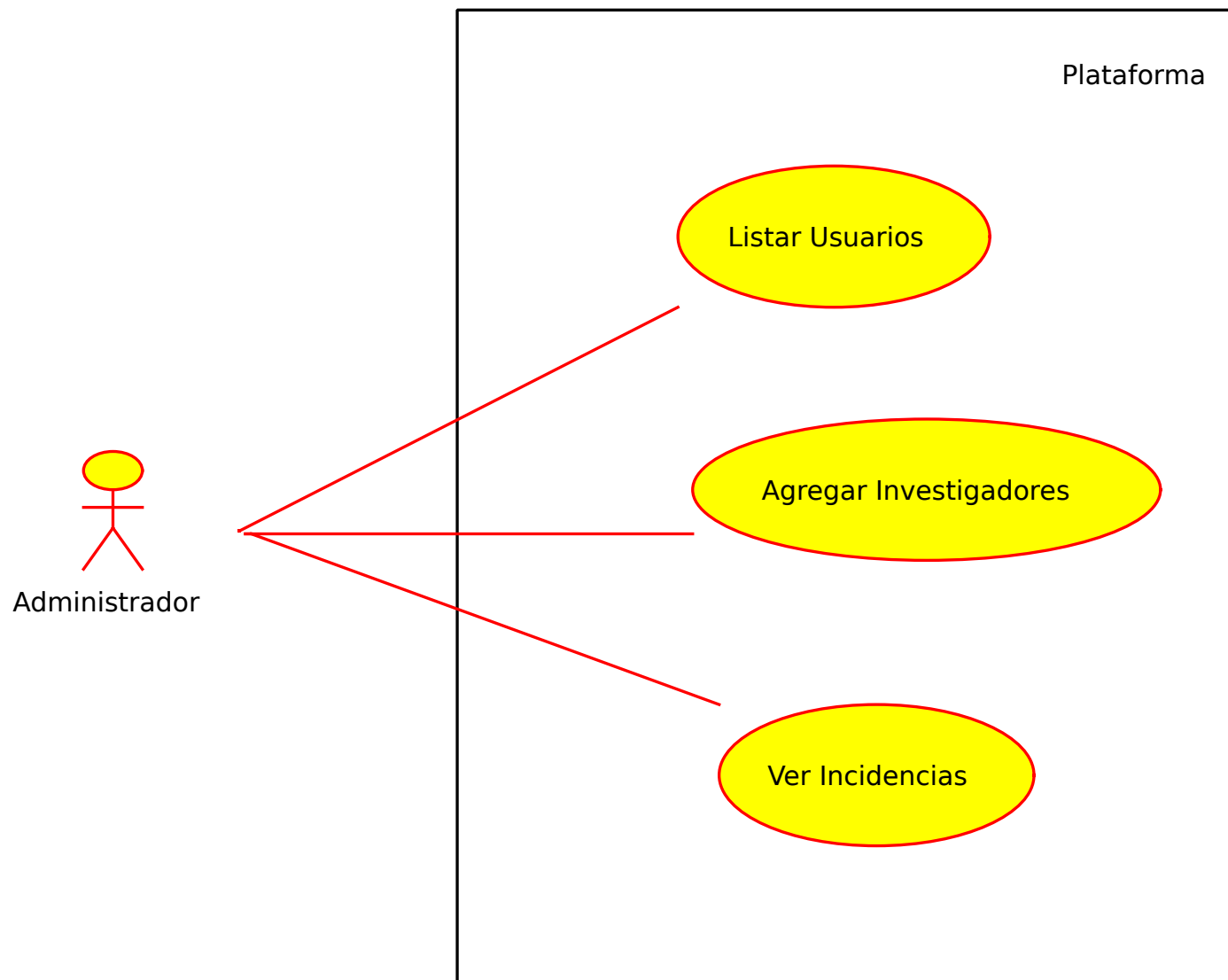


Figure E.9: Diagrama de casos del administrador

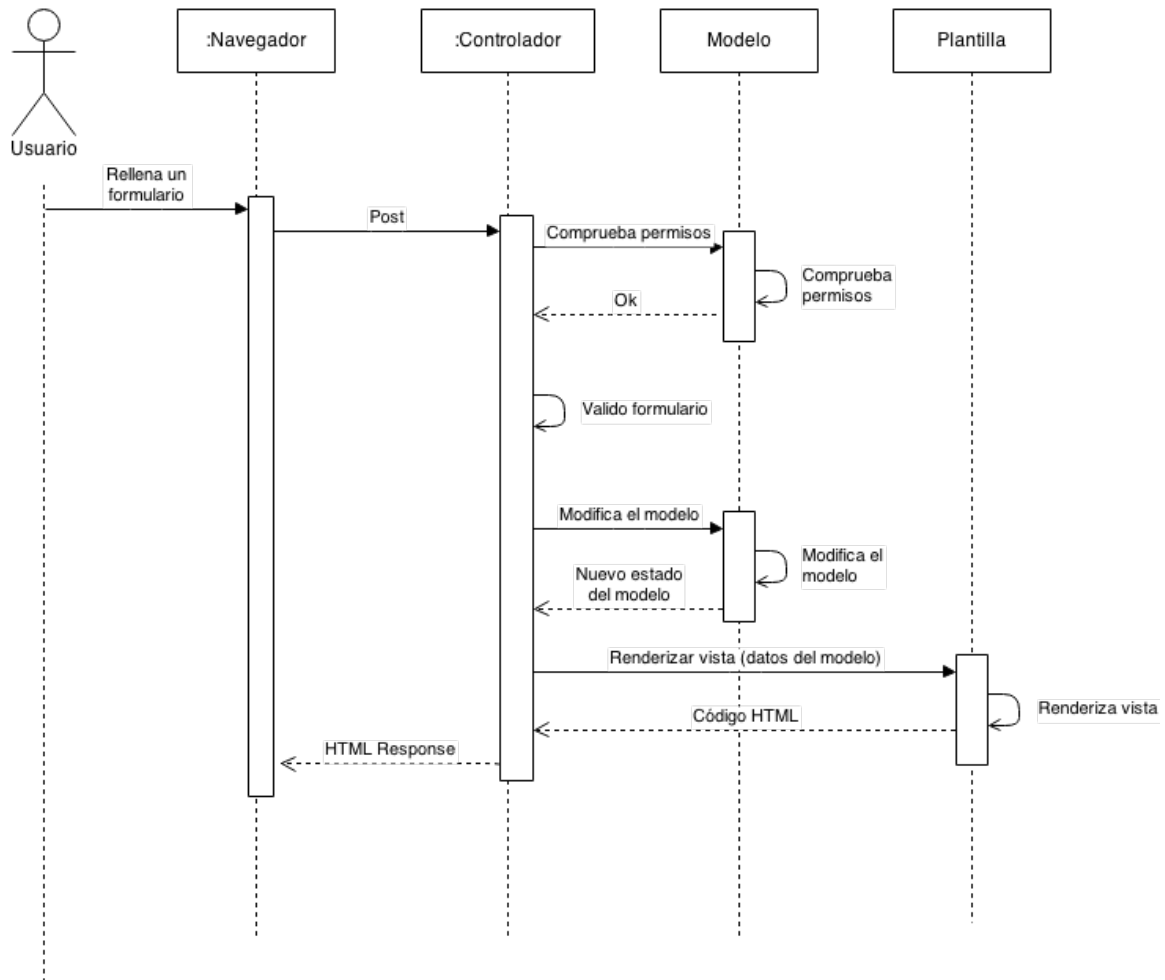


Figure E.10: Ejemplo de traza de eventos genérico

esta información es enviada al controlador de la aplicación, que procede a comprobar los permisos del usuario y validar el formulario. Una vez que es valido se modifica el modelo con los nuevos datos y se devuelve una nueva vista al usuario.

E.4. Módulos de la plataforma

Para la elaboración de las plataformas se ha seguido las recomendaciones dadas por los desarrolladores de Flask:

<http://flask.pocoo.org/docs/becomingbig/>

<https://github.com/mitsuhiko/flask/wiki/Large-app-how-to>

http://mattupstate.com/python/2013/06/26/how-i-structure-my-flask-applications.html?utm_

Como se puede ver en la figura E.11, se ha dividido la plataforma en distintos módulos, agrupando estas a nivel de funcionalidades.

E. DISEÑO DE LA APLICACIÓN

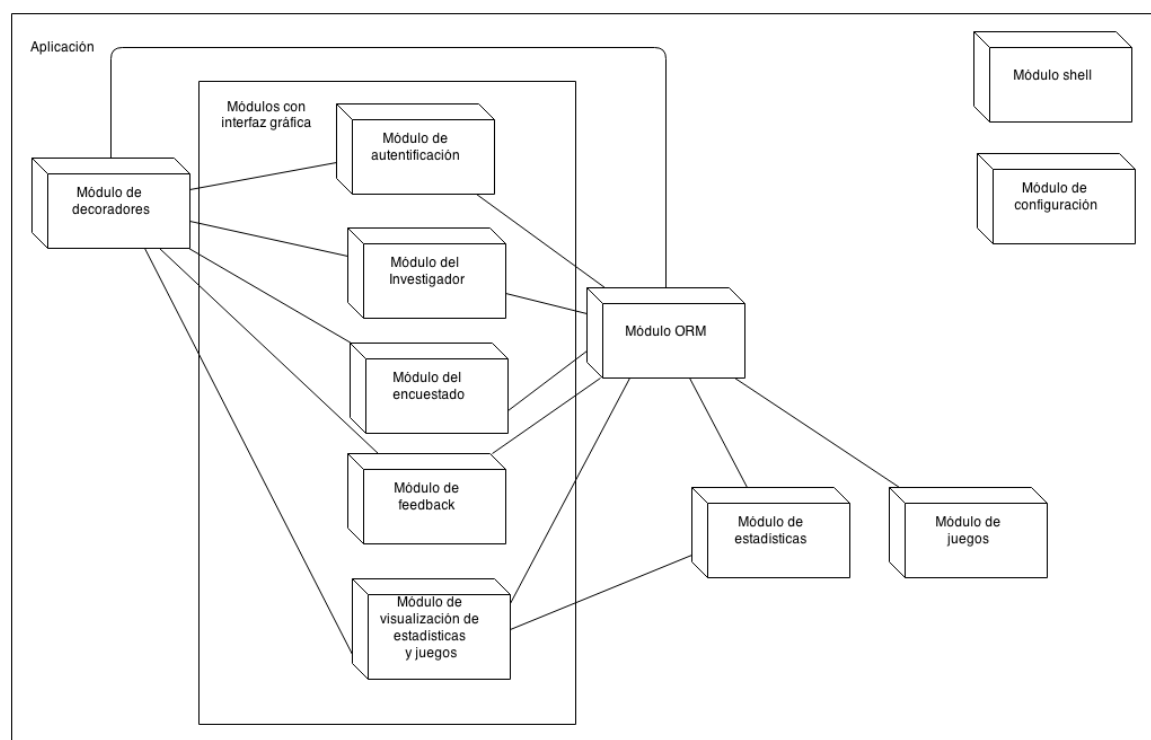


Figure E.11: Módulos del sistema

Los módulos con lo que se puede interactuar visualmente a través de un navegador tienen la siguiente estructura:

- **Vistas:** La vista de una función es donde una petición HTTP cumple con la lógica del programa. Es decir cada función tiene definida una ruta web por la que se puede acceder. En un modelo MVC, sería parte del controlador. Es esta función es donde los datos de la petición HTTP se traducen a una llamada a la lógica de la aplicación produciendo una respuesta.
- **Formularios:** Es el lugar donde están definidas las clases que contienen los formularios de un módulo con los que puede interactuar un usuario, así como las distintas validaciones que sufren los formularios.
- **Plantillas:** Cuando la vista genera una respuesta, es la plantilla quien se encarga de generar el código HTML y JavaScript, para representar los resultados al usuario.

E.4.1. Módulo de decoradores

Este módulo contiene los decoradores usados para proteger las vistas de los distintos módulos que se pueden acceder vía petición HTML, así se evita que cualquier

```
1 @blueprint.route('/survey/new', methods = ['GET', 'POST'])
2 @login_required
3 @researcher_required
4 def new_survey():
```

Figure E.12: Protección de una vista

usuario pueda ejecutar una vista a la que no tiene acceso. Esto se ha realizado mediante decoradores, ya que permiten alternar de manera dinámica la funcionalidad de una función, método o clase, sin tener que realizar ningún cambio de código en la clase decorada.

Los decoradores disponibles son los siguientes:

- **login_required**: Comprueba que el usuario ha ingresado en la plataforma.
- **researcher_required**: Comprueba que el usuario es un investigador.
- **finished_survey**: Comprueba que el usuario ha finalizado la encuesta.
- **belong_researcher**: Valida que la encuesta, sección o pregunta que quiere modificar pertenece al investigador en cuestión y no a otro.
- **valid_survey**: Comprueba que la encuesta ya se ha publicado.
- **there_is_stateSurvey**: Valida que existe y es válido el estado de la encuesta por parte de un usuario que quiere rellenar una encuesta.

Por ejemplo en la figura E.12, se puede comprobar que para acceder a la vista *new_survey*, que sirve para la creación de una nueva encuesta, debemos acceder a la dirección *http://servidor/survey/new*, pero antes de poder acceder, se comprueba que el usuario se haya autenticado en el sistema, mediante el decorador *@login_required*, y que además el usuario tenga permisos de investigador, *@researcher_required*. De esta forma se puede proteger las páginas a los que los usuarios pueden acceder.

E.4.2. Módulo Planificador de tareas.

Este módulo se encarga de ejecutar tareas periódicas. Concretamente tiene definidas dos:

- Envía periódicamente al investigador los resultados de la encuesta.
- En la adaptación del experimento "*¿Cómo son nuestros voluntarios?*", se decidió realizar los pagos manualmente, por lo que se envía un correo al responsable de los pagos, con el dinero y usuario al que debe realizar el pago.

E.4.3. Módulo de Autenticación

En el módulo de autenticación permite las siguientes opciones:

- Registrarnos en la plataforma: Uno se puede registrar mediante el uso de un correo.

A esta función se accede mediante la vista: *"auth/register"*.

- Validarnos mediante el uso de un correo y contraseña.

A esta función se accede mediante la vista: *"auth/loginEmail"*.

- Validarnos mediante el uso de un servidor OpenID.

A esta función se accede mediante la vista: *"auth/"* o *"auth/login"*.

- Cerrar la sesión actual.

A esta función se accede mediante la vista: *"auth/logout"*.

E.4.4. Módulo del investigador

Este módulo permite a los usuarios que sean investigadores, las siguientes acciones:

- Listado de encuestas creadas por un investigador.
 - A esta función se accede mediante las vistas: *"researcher/"* o *"researcher/index"*.
- Creación de encuestas: Durante la creación de una encuesta se comprueba que todos los campos del formulario son validos, además permite definir las siguientes opciones:
 - Título.
 - Descripción: Se hace uso de la sintaxis Markdown para el enriquecimiento del texto.
 - Número máximo de usuarios que pueden acceder a la encuesta, si es que lo hay.
 - Fecha de inicio de la encuesta.
 - Fecha de finalización de la encuesta.
 - Tiempo máximo para la realización de la encuesta, si es que lo hay.

A esta función se accede mediante la vista: `"researcher/survey/new"`

- **Modificación de una encuesta:** Se permite modificar cualquier campo de una encuesta creada.

A esta función se accede mediante la vista: `"researcher/survey/<int:id_survey>"`, siendo `id_survey` el identificador de la encuesta a modificar.

- **Eliminación de una encuesta:** Se permite eliminar una encuesta creada, siempre que no haya sido publicada.

A esta función se accede mediante la vista: `"researcher/survey/deleteSurvey/<int:id_survey>"` siendo `id_survey` el identificador de la encuesta a eliminar.

- **Creación de consentimientos:** Mediante un formulario, el investigador informa de los consentimientos que debe aceptar el usuario para poder participar en la encuesta. Hace uso de la sintaxis Markdown para el enriquecimiento del texto.

A esta función se accede mediante la vista: `"researcher/survey/<int:id_survey>/consent/add"` siendo `id_survey` el identificador de la encuesta.

- **Modificación de un consentimiento.**

A esta función se accede mediante la vista: `"researcher/survey/<int:id_survey>/consent/<int:id_consent>"` siendo `id_survey` el identificador de la encuesta, e `id_consent` el identificador del consentimiento

- **Eliminación de un consentimiento.**

A esta función se accede mediante la vista: `"researcher/survey/<int:id_survey>/consent/<int:id_consent>"` siendo `id_survey` el identificador de la encuesta, e `id_consent` el identificador del consentimiento

- **Creación de secciones y subsecciones:** El investigador puede definir con total libertad el número de secciones y subsecciones que tiene una encuesta, además permite definir las siguientes opciones:

- Título de la sección.
- Descripción: Se hace uso de la sintaxis Markdown para el enriquecimiento del texto.
- Secuencia de la sección.
- Porcentaje de número de usuarios que realizan la sección.

E. DISEÑO DE LA APLICACIÓN

El usuario que realice la encuesta recorrerá en profundidad el árbol generado por la encuesta. Si dos o mas ramas tienen la misma secuencia, se elige al azar que rama recorrer primero. Si dos o mas ramas con la misma secuencia, tienen un porcentaje menor que 1, este porcentaje indica la probabilidad que hay para recorrer esa rama y no cualquiera de las otras que tienen la misma secuencia.

A la función de añadir una sección a una encuesta dada, se accede mediante la vista: *"researcher/survey/<int:id_survey>/section/new"*, siendo *id_survey* el identificador de la encuesta.

A la función de añadir una sección a una sección dada, se accede mediante la vista: *"researcher/survey/<int:id_survey>/section/<int:id_section>/new"*, siendo *id_survey* el identificador de la encuesta, e *id_sección* el identificador de la sección de la que va a colgar la nueva sección.

- Modificación de una sección o subsección.

A esta función se accede mediante la vista: *"researcher/survey/<int:id_survey>/section/<int:id_section>/edit"*, siendo *id_survey* el identificador de la encuesta, e *id_sección* el identificador de la sección a modificar

- Eliminación de una sección o subsección:

A esta función se accede mediante la vista: *"researcher/survey/<int:id_survey>/deleteSection/<int:id_section>"*, siendo *id_survey* el identificador de la encuesta, e *id_sección* el identificador de la sección a eliminar.

- Creación de preguntas: Dada una sección, el investigador puede añadir las preguntas que desee a la sección. Las posibilidades que tiene son las siguientes:

- Preguntas cuya respuesta es Si o No, las opciones disponibles son las siguientes:
 - Texto de la pregunta: Se hace uso de la sintaxis Markdown para el enriquecimiento del texto.
 - Indicar si es una pregunta cuya respuesta es obligatoria o no.
 - Respuesta correcta a la pregunta si es que la tiene, sirve para crear preguntas de control.
 - Número máximo de intentos que tiene el usuario para responder bien a la pregunta.

- Preguntas cuya respuesta es un texto, las opciones disponibles son las siguientes:
 - Texto de la pregunta: Se hace uso de la sintaxis Markdown para el enriquecimiento del texto.
 - Indicar si es una pregunta cuya respuesta es obligatoria o no.
 - Respuesta correcta a la pregunta si es que la tiene, sirve para crear preguntas de control
 - Número máximo de intentos que tiene el usuario para responder bien a la pregunta.
 - Validar que el texto introducido por el usuario en la respuesta es:
 - ◇ Número entero.
 - ◇ Número decimal.
 - ◇ Comprobarlo mediante una expresión regular definido en sintaxis Python.
 - Mostrar un mensaje de error personalizado si ha introducido en un formato incorrecto la respuesta.
- Preguntas cuya respuesta es una opción dada, las opciones disponibles son las siguientes.
 - Texto de la pregunta: Se hace uso de la sintaxis Markdown para el enriquecimiento del texto.
 - Indicar si es una pregunta cuya respuesta es obligatoria o no.
 - Respuesta correcta a la pregunta si es que la tiene, sirve para crear preguntas de control.
 - Número máximo de intentos que tiene el usuario para responder bien a la pregunta.
 - Formato en el que se mostrará la pregunta, hay tres posibilidades:
 - ◇ Horizontal: Se mostrará las opciones mediante un radio botón, en una disposición horizontal.
 - ◇ Vertical: Se mostrará las opciones mediante un radio botón, en una disposición vertical.
 - ◇ Select: Se mostrará las opciones mediante un formulario de selección.

E. DISEÑO DE LA APLICACIÓN

- Definición de las respuestas disponibles mediante la definición de un rango de números, así como el salto entre ellos.
 - Definición de las respuestas disponibles añadiéndolas de una en una.
- Preguntas cuya respuesta es una escala likert, las opciones disponibles son las siguientes.
 - Texto de la pregunta: Se hace uso de la sintaxis Markdown para el enriquecimiento del texto.
 - Indicar si es una pregunta cuya respuesta es obligatoria o no.
 - Indicar el rango de la escala likert, que estará definida entre 0/1 y un número mayor que 1.
 - Etiqueta opcional para indicar el significado mínimo de la escala.
 - Etiqueta opcional para indicar el significado máximo de la escala

Todas las preguntas descritas anteriormente se pueden mostrar o no dependiendo de la respuesta dada a una pregunta de la misma sección.

A esta función se accede mediante la vista: *"researcher/survey/<int:id_survey>/section/<int:id_sección>"* siendo *id_survey* el identificador de la encuesta e *id_sección* el identificador de la sección donde se encuentra la pregunta.

■ Modificación de una pregunta:

A esta función se accede mediante la vista: *"researcher/survey/<int:id_survey>/section/<int:id_sección>/question/<int:id_pregunta>"* siendo *id_survey* el identificador de la encuesta e *id_sección* el identificador de la sección donde se encuentra la pregunta, e *id_pregunta* el identificador de la pregunta a modificar

■ Eliminación de una pregunta:

A esta función se accede mediante la vista: *"researcher/survey/<int:id_survey>/Section/<int:id_sección>/question/<int:id_pregunta>"* siendo *id_survey* el identificador de la encuesta e *id_sección* el identificador de la sección donde se encuentra la pregunta, e *id_pregunta* el identificador de la pregunta a eliminar

■ Duplicar sección: Permite duplicar una sección así como todas las secciones y preguntas que cuelgan de su rama.

A esta función se accede mediante la vista:

- *"researcher/survey/<int:id_survey>/duplicateSection/<int:id_section>/section/<int:id_section2>"*, siendo *id_survey* el identificador de la encuesta, *id_sección* el identificador de la sección a copiar, *id_section2* el identificador de la sección donde se copiará.
- *"researcher/survey/<int:id_survey>/duplicateSection/<int:id_section>/survey/"*, siendo *id_survey* el identificador de la encuesta, *id_sección* el identificador de la sección a copiar, se copiará directamente sobre la raíz, la cual es la encuesta.

- Exportar una encuesta: Permite exportar a un fichero XML, la encuesta seleccionada.

A esta función se accede mediante la vista: *"researcher/survey/exportSurvey/<int:id_survey>"* siendo *id_survey* la encuesta a exportar

- Importar una encuesta: Permite importar una encuesta mediante un fichero XML.

A esta función se accede mediante la vista: *"researcher/survey/new"*

- Exportar estadísticas de una encuesta: Permite exportar en un fichero CSV, los resultados de la encuesta seleccionada.

A esta función se accede mediante la vista: *"researcher/survey/exportStats/<int:id_survey>"* siendo *id_survey* la encuesta a exportar

Las vistas anteriores solo están disponibles para investigadores, además no se pueden editar encuestas de otros investigadores, ya que todas las vistas están protegidas mediante el decorador *"@belong_researcher"*.

E.4.5. Módulo del encuestado

Este módulo permite a los usuarios que sean voluntarios realizar rellenar encuestas. Las acciones disponibles son las siguientes:

- Listar todas las encuestas disponibles, así como el estado de esta, ya sea sin empezar, empezada o finalizada.

A esta función se accede mediante la vista: *"survey/index"* o *"survey/"*

E. DISEÑO DE LA APLICACIÓN

- Empezar o continuar una encuesta: Es la función encargada de decidir cual es el siguiente paso cuando se realiza una encuesta, ya sea mostrar los consentimientos, mostrar una sección de la encuesta, indicar que se ha finalizado la encuesta o mostrar el feedback de la encuesta si es que la tiene.

A esta función se accede mediante la vista: `"survey/<int:id_survey>"`, siendo `id_survey` la encuesta a realizar.

- Mostrar los consentimientos: Esta función es la encargada de mostrar el consentimiento que el usuario debe aceptar para realizar la encuesta.

A esta función se accede mediante la vista `"survey/<int:id_survey>/consent"`, siendo `id_survey` la encuesta a realizar

- Mostrar una sección: Esta función es la encargada de mostrar una sección, con todas las preguntas que hay en ella, además de guardar los resultados de las preguntas, así como los tiempos empleados en ellas.

A esta función se accede mediante las vistas `"survey/<int:id_survey>/section/<int:id_section>"` o `"survey/<int:id_survey>/section/"`, siendo `id_survey` la encuesta a realizar e `id_section`, la sección.

Todas las vistas están protegidas, de manera que es obligatorio haber ingresado en la plataforma, además en la vista *Mostrar una sección* solo se permite acceder a la sección que le toca realizar al usuario.

E.4.6. Módulo de Feedback

Este módulo permite dar feedback sobre las decisiones tomadas en el proyecto *"¿Cómo son nuestros voluntarios?"*. Cada vista corresponde con una decisión concreta del experimento.

- Lógica: Durante el desarrollo del proyecto *"¿Cómo son nuestros voluntarios?"*, se decidió que el orden del feedback sobre las decisiones tomadas, iba a ser el mismo que el realizado durante el experimento, y debido a que cada voluntario se le asigna un orden al azar, esta función es la encargada de decidir que feedback se le va a mostrar al voluntario, siempre siguiendo el mismo orden de la encuesta.

A esta función se accede mediante la vista `"feedback/id_survey"`, siendo `id_survey` la encuesta de la que recibir feedback.

- Juegos de la lotería, la renta y dictador: Corresponde a las decisiones de las versiones de los dos juegos de la lotería y los dos juegos de la renta. Muestra la respuesta del usuario y la media dada por los demás voluntarios, además muestra el porcentaje de los voluntarios que han tomado la misma decisión que el usuario.
- Juego del ultimatum: A parte de mostrar la información que se da en los demás juegos, también se muestra el porcentaje de voluntarios que hubiesen aceptado la cantidad dada por el usuario.

E.4.7. Módulo de estadísticas:

Este módulo es el encargado de generar el fichero de resultados para los investigadores. Concretamente existen dos funciones, una para encuestas genéricas y otra para el experimento "*¿Cómo son nuestros voluntarios?*". Ya que se adapto los resultados a la necesidad de los investigadores en cuanto al formato requerido por ellos para trabajar más cómodamente.

E.4.8. Módulo Juegos:

Este módulo es el encargado de decidir en los juegos del experimento "*¿Cómo son nuestros voluntarios?*", que usuarios se enfrentan con quién. Además de buscar las respuestas dadas por los voluntarios.

A la hora de decidir que usuarios van a enfrentarse, se busca a todos los usuarios que no han realizado ningún enfrentamiento. Si este llega a un número mínimo necesario para que no existan enfrentamientos repetidos, se continua. Sino se selecciona las decisiones tomadas de otros voluntarios que ya se han enfrentado, pero sin posibilidad de que estos últimos vuelvan a salir premiados.

E.4.9. Módulo de visualización de estadísticas y juegos.

Este módulo permite la visualización de los resultados de las encuestas, además también permite la visualización de los resultados de los juegos del experimento "*¿Cómo son nuestros voluntarios?*"

- Listar todas las encuestas creadas por un investigador.

A esta función se accede mediante la vista "*stats/*" o "*stats/index*"

E. DISEÑO DE LA APLICACIÓN

- Listar el resultado de una encuesta: Representa mediante una tabla el fichero CSV de las encuestas, se puede elegir el criterio de ordenación.

A esta función se accede mediante la vista `"stats/<int:id_survey>"`, siendo `id_survey` la encuesta de la que se desea visualizar los resultados.

- Listar los juegos disponibles en una encuesta:

A esta función se accede mediante la vista `"stats/<int:id_survey>/games"`, siendo `id_survey` el identificador de la encuesta

- Listar el resultado de un juego de una encuesta: Representa mediante una tabla, los resultados de de los distintos juegos de una encuesta, se puede elegir el criterio de ordenación.

A esta función se accede mediante la vista `"stats/<int:id_survey>/games/gameX"`, siendo `id_survey` el identificador de la encuesta.

Las vistas anteriores solo están disponibles para investigadores, además solo se puede acceder a las encuestas creadas por el propio investigador.

E.4.10. Módulo de modelo

Este es el módulo descrito anteriormente en la sección E.1

Casi todos los demás módulos interaccionan con este Módulo.

E.4.11. Módulo de configuración

Este módulo se encarga de leer la configuración de la plataforma una vez que se inicia está. Aunque no se ha señalado explícitamente, los demás módulos interaccionan con él si tienen que leer alguna configuración de la plataforma.

Se han tenido en cuenta las recomendaciones dadas en <http://flask.pocoo.org/docs/config/#configuration-best-practices>

Y se ha decidido crear un sistema de clases y herencia para la configuración del sistema, pudiendo cambiar el modo de funcionamiento de la aplicación simplemente cambiando la variable de entorno `FLASK_CONFIG`, la estructura que sigue y los modos disponibles son los siguientes:

```
1 class Config(object):
2     # configuración básica de la plataforma
3
4 class DevelopmentConfig(Config):
5     # Configuración de la plataforma en modo desarrollo
6     DEBUG = True
7
```

```
8 class TestingConfig(Config):
9     # Configuración de la plataforma para test unitarios
10
11 class Jmeter(Config):
12     # Configuración de la plataforma para la realización de pruebas de sobrecarga
13
14 class ProductionConfig(Config):
15     # Configuración de la plataforma en modo producción
16     # Se define quien es el usuario administrador y cual es el servidor de correo
17     # para enviar los distintos logs
18
19 class HerokuConfig(ProductionConfig):
20     # Configuración de la plataforma en modo producción en la nube Heroku
21     # Usa el sistema de logs de Heroku para almacenarlos.
22
23 class UnixConfig(ProductionConfig):
24     # Configuración de la plataforma en modo producción en una máquina de tipo Unix
25     # Hace uso del sistema de logs de la máquina Unix.
```

Por otra parte, para facilitar el tener varias configuraciones posibles, se carga primeramente la configuración básica definida anteriormente y luego se puede reemplaza por la leída en la variable de entorno `SWARMS_SURVEY_SETTINGS`, pudiendo cambiar fácilmente la configuración.

E.4.12. Módulo shell:

Este módulo se accede mediante la consola de la plataforma, dando acceso a un interprete Python.

```
1 (venv)$ ./manage.py shell
2 // shell interactively of swarm-surveys, "import utiles" to access the administration
   tools
3 In [1]: from current_app import utiles
```

Entre las opciones disponible en *utiles*, está la de listar usuarios, dar o quitar permisos de investigador a un usuario dado o reiniciar los juegos del experimento *"¿Cómo son nuestros voluntarios?"*

Además mediante el uso de esta interfaz de linea de comandos, se puede acceder a cualquier módulo definido en la plataforma.

Por ejemplo, ejecutar la función de generar los resultados de una encuesta:

```
1 In [1]: from current_app import models
2 In [2]: from current_app import stats
3 In [2]: survey = models.Survey.query.get(1)
4 In [3]: stats.write_stats(survey)
5 written file in /home/jarenere/swarm-surveys/stats_csv/¿Cómo_son_nuestros_voluntarios
   ?_1.csv
```

E.5. Prototipado de las ventanas y cuestiones de usabilidad

Después del análisis y el diseño de las distintas partes de la aplicación, se continuo con un diseño preliminar de la interfaz para el usuario. Se tomaron distintas ideas de las plataformas estudiadas para la elaboración de encuestas, así como las guías de usabilidad del proyecto KDE, <http://techbase.kde.org/Projects/Usability/HIG>

Para facilitar al investigador la navegación entre las distintas secciones, se decidió mostrar el árbol de secciones por completo en un lateral de la ventana. Además de mostrar en la parte superior un menú de tipo "migas de pan", que presenta en forma textual todos los enlaces que describen la ruta de una sección a partir de la raíz, siendo esta la propia encuesta.

Por otra parte para mejorar la usabilidad de la aplicación, se ha añadido código JavaScript en distintas páginas para ofrecer ciertas funcionalidades al usuario, sin tener que realizar peticiones a la plataforma web, añadiendo velocidad y fluidez a la plataforma.

Se decidió también añadir un editor y visualizador de código Markdown para poder comprobar en tiempo real el formato de texto que verían los usuarios a la hora de realizar encuestas.

Por otra parte para mejorar la usabilidad de la plataforma, se uso un sistema de URL amigables, para recordar mas fácilmente la funcionalidad de cada página web.

Además durante el prototipado de las ventanas, se tuvo en cuenta las funcionalidades que provee el framework Bootstrap, para facilitar que las distintas páginas se adaptasen al tamaño del navegador.

E.5.1. Diseño del experimento "¿Cómo son nuestros voluntarios?"

Debido a la naturaleza del proyecto, siempre se intenta evitar el influir a los participantes, para que las respuestas que den sean las que realmente quieren dar. Esto incluye cualquier tipo de feedback visual, incluso el nombre del proyecto "¿Cómo son nuestros voluntarios?", induce a pensar como voluntario.

El diseño final para el experimento, se puede ver en las siguientes capturas

E.5 Prototipado de las ventanas y cuestiones de usabilidad

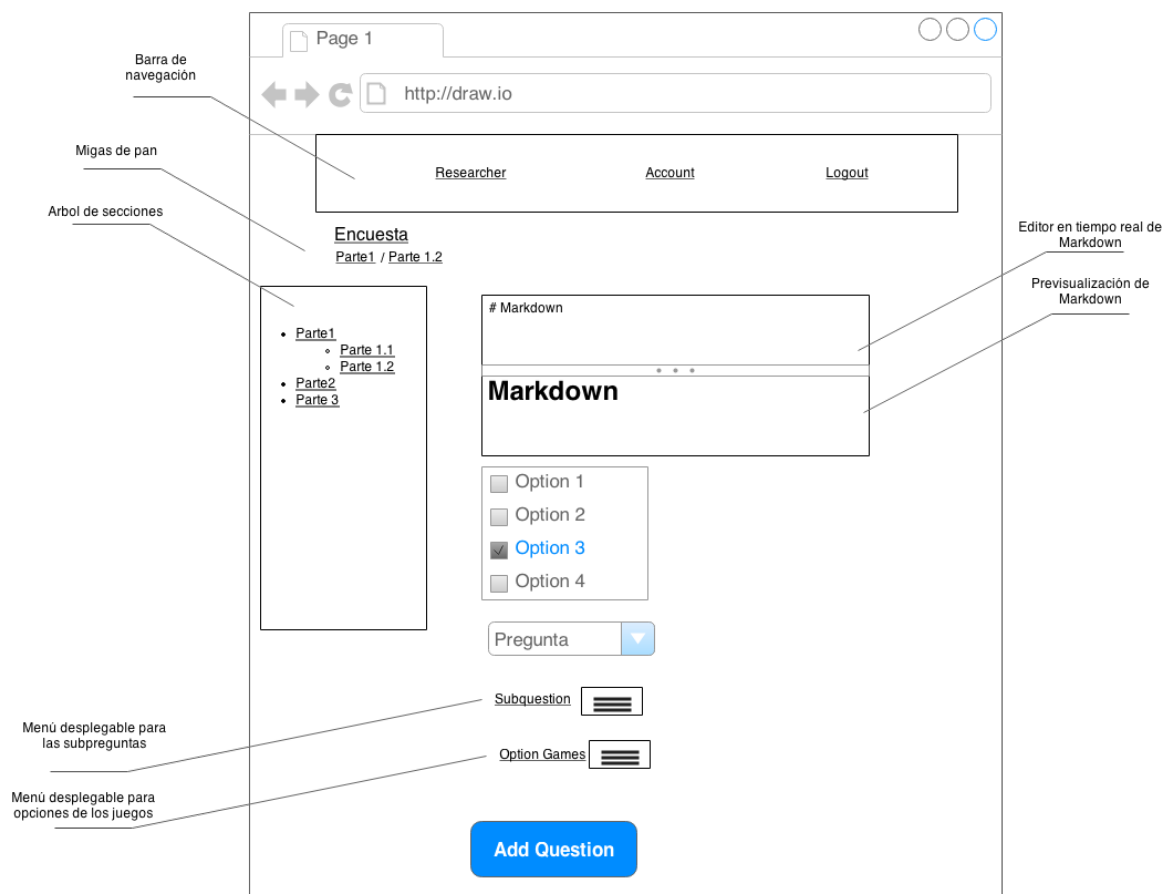


Figure E.13: Prototipo de creación de una nueva pregunta

E. DISEÑO DE LA APLICACIÓN

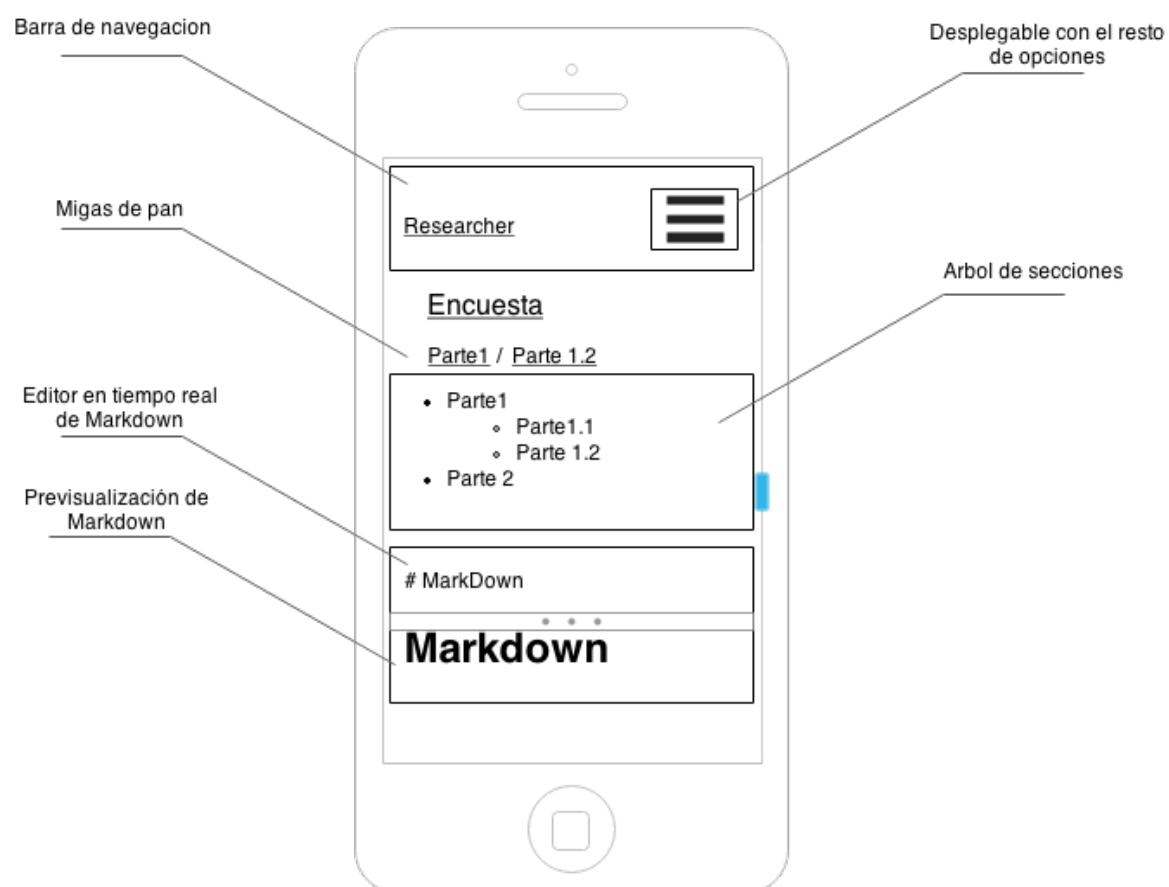


Figure E.14: Prototipo móvil de creación de una nueva pregunta

[Swarms game theory](#) [Researcher](#) [Cuenta](#) [Cerrar sesión](#)

Encuesta

[Parte 1](#) / [Parte 1.1](#) /

Add Question!!

- [Parte 1](#)
 - [Parte 1.1](#)
 - [Parte 1.2](#)
- [Parte 2](#)

Text

```
## This is a example of Markdown  
**Markdown** is rendered on the fly in the <i>preview area</i>!  
  
More about [markdown](http://daringfireball.net/projects/markdown/).
```

This is a example of Markdown

Markdown is rendered on the fly in the *preview area*!

More about [markdown](#).

Required ☒

Type of question

answer

Number of attempt

SubQuestion

Options game

Figure E.15: Ventana de creación de una nueva pregunta

The image shows a web form for creating a new question. It is divided into two main sections: 'SubQuestion' and 'Options game'. The 'SubQuestion' section contains three fields: 'Type of operation' with a dropdown menu set to 'None', 'Value' with a text input field, and 'Question' with a dropdown menu. The 'Options game' section contains two fields: 'Type of question' with a dropdown menu set to 'None', and 'It is with money real' with an unchecked checkbox. At the bottom, there is a 'feedback' checkbox, also unchecked.

SubQuestion

Type of operation: None

Value:

Question:

Options game

Type of question: None

It is with money real: ☐

feedback: ☐

Figure E.16: Despliegue de opciones de creación de una nueva pregunta

Swarms game theory

Encuesta

Parte 1 / Parte 1.1 /

Add Question!!

- [Parte 1](#)
 - [Parte 1.1](#)
 - [Parte 1.2](#)
- [Parte 2](#)

Text

This is a example of Markdown
Markdown is rendered on the fly in the <i>preview area</i>!
More about [markdown]

This is a example of Markdown
Markdown is rendered on the fly in the *preview area*!
More about [markdown](#).

Required

☒

Type of question

YES/NO

Figure E.17: Visualización móvil

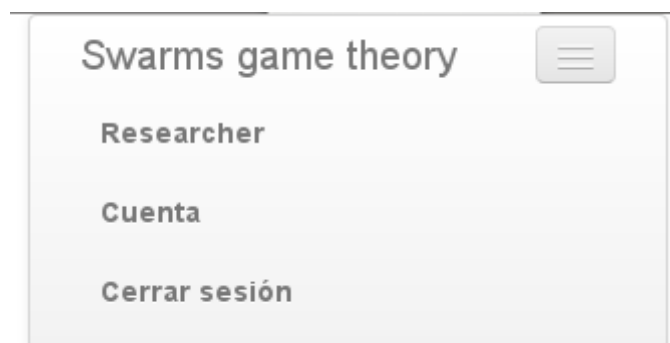


Figure E.18: Menú desplegado

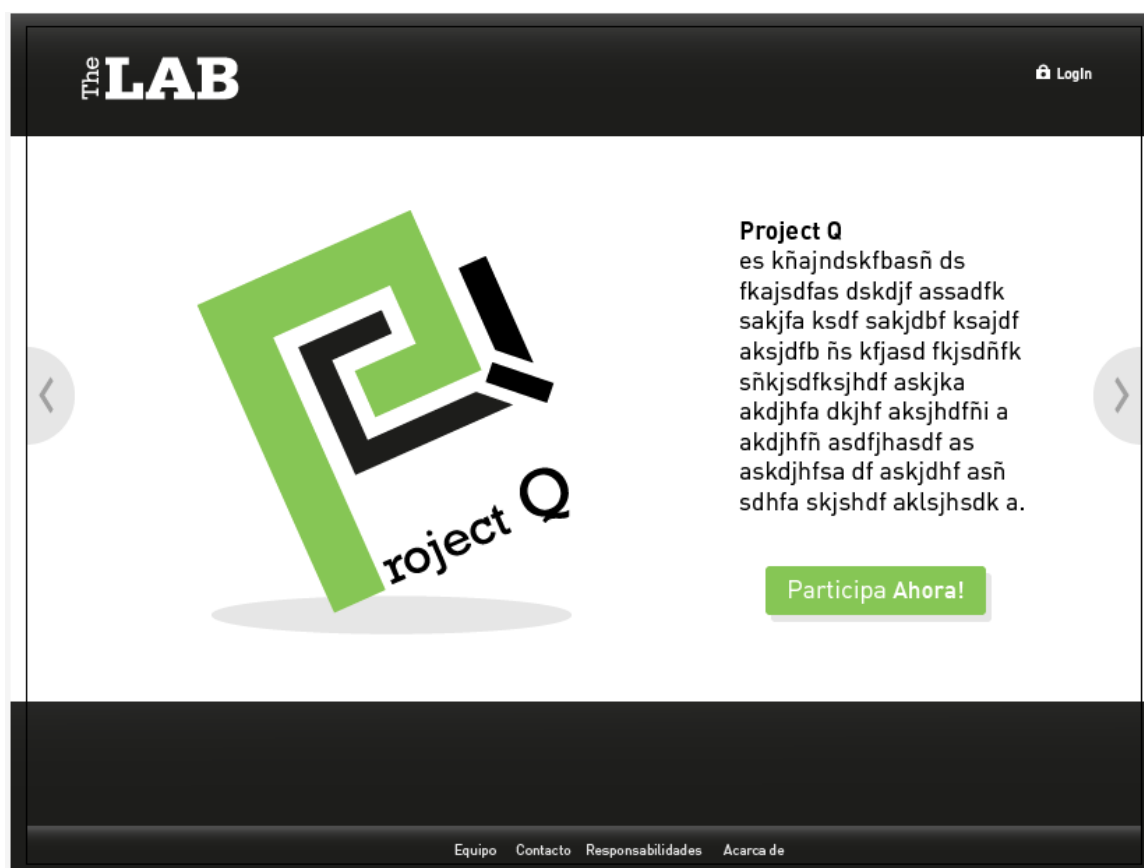


Figure E.19: Visualización experimento "¿Cómo son nuestros voluntarios?"

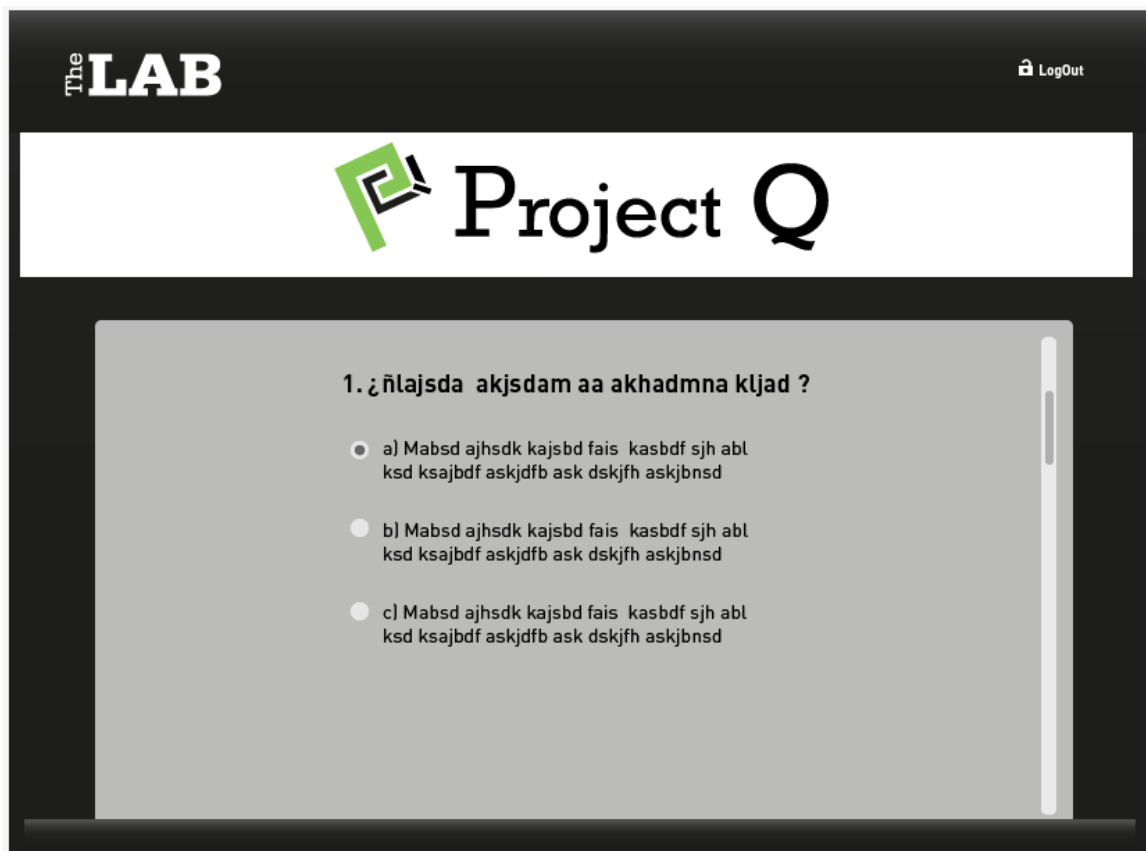


Figure E.20: Visualización experimento "¿Cómo son nuestros voluntarios?"

E. DISEÑO DE LA APLICACIÓN

Appendix F

¿Cómo son nuestros voluntarios?

Este anexo incluye parcialmente el documento de partida que se usó para realizar el experimento y la plataforma. En las decisiones 1, 2 y 3 de la parte 3 solo se ha incluido el texto de los distintos juegos y las preguntas de la decisión 1 debido a su similitud.

F.1. Proyecto "¿Cómo son nuestros voluntarios?"

- Antonio Espín (Universidad de Granada)
- Pablo Brañas-Garza (Middlesex University)
- Anxo Sánchez (Universidad Carlos III)
- Equipo Ibercivis

F.2. Introducción y motivación

El proyecto se propone caracterizar de manera detallada el perfil de los voluntarios que toman parte en los proyectos de ciencia ciudadana de Ibercivis. El punto de partida es el trabajo "Experimental subjects are not different" [F. Exadaktylos, A. M. Espín & P. Brañas-Garza, Sci. Rep. 3, 1213 (2013)], donde se comparó a los sujetos experimentales típicos de los laboratorios de economía con la población general. Los resultados de la acción en Ibercivis permitirían:

- Comparar a los voluntarios de Ibercivis con la muestra de población general del trabajo citado y analizar su representatividad
- Caracterizar a una población amplia de voluntarios para después solicitar su colaboración en futuros proyectos seleccionándolos conforme a su perfil

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

F.3. Ejecución del proyecto

Nos ocupamos aquí de la implementación de la primera fase, la realización de una encuesta con incentivos económicos para que las decisiones tomadas se correspondan con las que se esperan de la teoría económica. De cara a la segunda fase, sería necesario almacenar las respuestas de los voluntarios de manera que fuera compatible con la preservación de su anonimidad y con la ley de protección de datos.

La encuesta que deben realizar los voluntarios se presenta en detalle a continuación, junto con las correspondientes observaciones sobre implementación. Por supuesto, todos aquellos puntos que no queden claros los resolveremos en conjunto, incluyendo posibles dificultades técnicas.

F.4. Encuesta

F.4.1. Instrucciones

Nota de los investigadores:

Estas instrucciones deben presentarse a los voluntarios al acceder a la aplicación. Deben terminar con una pregunta sobre si las han comprendido y sobre si aceptan las condiciones de participación. En particular, hay que asegurarse de que entienden que algunas cosas se pagarán y otras no, y si se les paga con qué probabilidad. Aquí como a lo largo del documento hay partes que deben aparecer en dos versiones; cada voluntario al llegar a cada uno de esos puntos es asignado a una de las dos versiones al azar, pero se debe registrar por cuales ha ido pasando.

Gracias por participar en este proyecto de investigación. Nuestro objetivo es entender cómo se toman las decisiones, y no esperamos ningún comportamiento en concreto por tu parte. Tus respuestas serán guardadas respetando en todo momento tu anonimato; los investigadores no tendrán acceso a tus datos personales ni podrán conectar sus respuestas con ellos de ninguna manera. Además, se te incluirá en la base de datos de Ibercivis para, llegado el caso, solicitar tu colaboración voluntaria en futuras investigaciones.

A continuación se te pasará a una encuesta, que consta de tres partes: una primera consistente en una batería de preguntas sobre distintos aspectos de tu personalidad, y dos partes en las que se te presentarán diversos escenarios en los que tendrás que tomar decisiones; en un caso serán decisiones que sólo dependen de ti, mientras que

en el otro dependerán también de lo que haga otro participante anónimo escogido al azar, que no te conoce y al que tú no conoces (y nunca se os revelarán vuestras identidades), y que puede haber escogido sus respuestas antes o después que tú. En la segunda y tercera partes, las cuestiones se plantean en términos de una ganancia económica. En algunos casos, esa ganancia económica es meramente hipotética; sin embargo, para el objetivo del proyecto, es importante que tomes tus decisiones como si realmente fueras a recibir ese dinero. En otros, uno de cada diez participantes en el proyecto, elegidos al azar, recibirá el pago asociado a una de sus decisiones, también escogida al azar. Al final de la encuesta se te comunicará si has sido seleccionado y en tal caso se te informará del procedimiento para el pago.

Tu participación en el proyecto te llevará entre media hora y una hora. Es deseable que participes sin interrupción, realizando la encuesta de principio a fin. Sin embargo, si no te fuera posible, podrás guardar tus respuestas hasta el punto al que hubieras llegado y terminarla después.

¿Aceptas las condiciones de participación? Sí (S), No (N)

¿Eres consciente de que no todas las decisiones de las partes segunda y tercera se pagarán con dinero real, y de que cuando lo vaya a ser se pagarán sólo a uno de cada diez participantes elegidos al azar? Sí (S), No (N)

Nota de los investigadores:

Esto exige el asignar un login al voluntario e implementar una manera de guardar las respuestas. El login puede ser también la manera de referirse a los voluntarios para futuras solicitudes de colaboración; los datos pueden estar asociados al login y podríamos tener una base de datos aparte donde los logins estén asociados a la persona, y a la que nosotros no tendríamos acceso. Tenemos que implementar además una manera de pagar. No son muchos, en principio se podría hacer manualmente, pero tenemos que verlo. Además de las respuestas, hay que guardar el momento en que van pinchando en cada opción, para luego analizar los tiempos que emplean en cada decisión, y poder identificar posibles respuestas al azar.

F.4.2. Primera parte

F.4.2.1. Bloque 1

1. Año de nacimiento
2. Localidad de nacimiento (y país si no es España)

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

3. ¿Hombre?: Sí (S), No (N)
4. Estado civil: soltero (1), casado (2), divorciado/separado (3), viudo (4), convive en pareja (5)
5. ¿Cuántos hijos has tenido?
 - 5.1 ¿Viven todos actualmente?: Sí (S), No (N)
6. Consideras que tu estado de salud es: muy malo (1), malo (2), regular (3), bueno (4), muy bueno (5)
7. ¿Eres fumador/a?: Sí (S), No (N)
 - 7.1 (Si fuma) Número medio de cigarros diarios
 - 7.2 (Si fuma) ¿Has intentado alguna vez de dejar de fumar?
 - 7.3 (Si lo ha intentado) ¿Cuántas veces?
8. ¿Tienes algún tipo de discapacidad reconocida?: no (0), entre 33% y 65% (1), más de 65% (2)
- ...
16. ¿Has cambiado tu denominación religiosa a lo largo de tu vida?: Sí (S), No (N)
 - 16.1 (Si ha cambiado) Antes te denominabas: (ídem 15)
 - 16.2 (Si ha cambiado) ¿A qué edad te cambiaste?
17. Crees que el éxito en la vida se debe principalmente a (sólo una opción):
 - a) La suerte
 - b) El esfuerzo

F.4.2.2. Bloque 2

Ahora tienes que responder si estás de acuerdo o no con las siguientes afirmaciones en una escala de 1 a 7. 1 significa totalmente en desacuerdo y 7 totalmente de acuerdo, el punto neutral es el 4.

Nota de los investigadores:

Para esto tenemos que acordar una forma visual para las likert (las escalas de 1 a 7) que nos guste y que sea cómoda para los voluntarios. Cada pregunta lleva una escala para contestar.

18. Estoy dispuesto a hacer un trabajo aburrido para devolver una ayuda previa.
19. Estoy dispuesto a dedicar tiempo y esfuerzo para devolver una injusticia que me hayan hecho
20. No me preocupa cuánto dinero tengo, lo que me preocupa es que otros tienen menos que yo

...

34. Ayudaría a un conocido que sé que no lo haría por mí

F.4.2.3. Bloque 3

Las siguientes preguntas tienen sólo dos posibles respuestas, hay que elegir una.

37. En general, ¿crees que se puede tener confianza en la mayoría de la gente, o que se debe ser muy prudente al relacionarse con la gente?
 - a) Se puede tener confianza en la mayoría de la gente.
 - b) Se debe ser muy prudente al relacionarse con la gente
38. ¿Crees que la mayoría de la gente intentaría aprovecharse de ti, si tuvieran la oportunidad, o tratarían de ser justos?
 - a) La mayoría de la gente intentaría aprovecharse de ti.
 - b) La mayoría de la gente tratarían de ser justos

...

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

F.4.2.4. Bloque 4

Las siguientes 4 preguntas, a diferencia de las demás, SÍ tienen una única respuesta correcta:

- 40. Si la probabilidad de contraer una enfermedad es de un 10 por ciento, ¿cuántas personas de 1.000 contraerían la enfermedad? (N si no puede/quiere responder)
- 41. Si 5 personas tienen el número premiado de la lotería y el premio a repartir es de dos millones de euros, ¿cuánto recibiría cada una?: (N si no puede/quiere responder)

...

F.4.2.5. Bloque 5

- 45. Lanzamos una moneda al aire. Elige, de entre estas dos opciones, la que prefieres:
 - a) Recibir 1.000 euros independientemente de si sale cara o sale cruz
 - b) Recibir 2.000 euros si sale cara y nada si sale cruz
- 46. Elige, de entre estas dos opciones, la que prefieres:
 - a) Recibir un billete de lotería con un 80% de probabilidad de ganar 45 euros y un 20% de probabilidad de no ganar nada
 - b) Recibir 30 euros
- 47. ¿Aceptarías el siguiente acuerdo? Lanzamos una moneda al aire y si sale cara ganas 1.500 euros, y si sale cruz pierdes 1.000 euros: Sí (S), No (N)

F.4.3. Segunda parte

Nota de los investigadores:

Queremos aleatorizar la presentación de las partes dos y tres. La mitad de los voluntarios hacen primero la segunda parte, detallada en esta sección y luego la tercera parte, detallada en la sección F.4.4, y la otra mitad al revés.

A su vez, la mitad de los encuestados realizaran esta parte con dinero real, y la otra mitad con dinero ficticio.

Además en esta parte queremos que en cada subtratamiento (2-3/3-2) la mitad hagan las cuestiones en orden 1-10, 11-20, y la otra mitad en el otro orden, 11-20, 1-10. La probabilidad de que se te presente un orden u otro es el 50% y debe ser independiente de que hagas 2-3 o 3-2. Además, la mitad tienen que hacer versión 1 (pago real) y la mitad versión 2 (pago hipotético). Cogiendo el ejemplo de los que hacen primero la parte 2 y después la 3 (igual para los que hacen 3-2), tendríamos 4 subgrupos: un 25% de gente haciendo la versión 1 en el orden 1-10, 11-20; otro 25% haciendo la versión 1 en orden 11-20, 1-10; otro 25% haciendo la versión 2 en orden 1-10, 11-20; y el 25% restante haciendo la versión 2 en orden 11-20, 1-10. Así que al final tendríamos 8 subgrupos (estos 4 más los 4 de los que hacen la parte 3 y luego la 2) con el mismo número de participantes aproximadamente.

F.4.3.1. Bloque 1

1. Recibir 30€ hoy o recibir 30€ dentro de un mes
2. Recibir 30€ hoy o recibir 32€ dentro de un mes
3. Recibir 30€ hoy o recibir 34€ dentro de un mes
4. Recibir 30€ hoy o recibir 36€ dentro de un mes
5. Recibir 30€ hoy o recibir 38€ dentro de un mes
6. Recibir 30€ hoy o recibir 40€ dentro de un mes
7. Recibir 30€ hoy o recibir 42€ dentro de un mes
8. Recibir 30€ hoy o recibir 44€ dentro de un mes
9. Recibir 30€ hoy o recibir 46€ dentro de un mes
10. Recibir 30€ hoy o recibir 48€ dentro de un mes

F.4.3.2. Bloque 2

11. Recibir 30€ dentro de un mes o recibir 30€ dentro de 7 meses
12. Recibir 30€ dentro de un mes o recibir 32€ dentro de 7 meses
13. Recibir 30€ dentro de un mes o recibir 34€ dentro de 7 meses

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

14. Recibir 30€ dentro de un mes o recibir 36€ dentro de 7 meses
15. Recibir 30€ dentro de un mes o recibir 38€ dentro de 7 meses
16. Recibir 30€ dentro de un mes o recibir 40€ dentro de 7 meses
17. Recibir 30€ dentro de un mes o recibir 42€ dentro de 7 meses
18. Recibir 30€ dentro de un mes o recibir 44€ dentro de 7 meses
19. Recibir 30€ dentro de un mes o recibir 46€ dentro de 7 meses
20. Recibir 30€ dentro de un mes o recibir 48€ dentro de 7 meses

F.4.4. Tercera parte

Nota de los investigadores:

En esta parte también queremos que en cada subtratamiento (2-3/3-2) la mitad hagan la versión 1 y la otra mitad la versión 2. La probabilidad de que se te presente una versión u otra es el 50% y debe ser independiente de que hagas 2-3 o 3-2. Además hay que aleatorizar el orden de las decisiones pero siempre manteniendo seguidas las decisiones 4 y 5 (ó 5 y 4, según el orden que salga aleatoriamente) que son los dos roles del mismo juego y las decisiones 2 y 3 (ó 3 y 2, según el orden que salga aleatoriamente).

A su vez, la mitad de los encuestados realizarán esta parte con dinero real, y la otra mitad con dinero ficticio.

F.4.4.1. Decisión 1

Versión 1 Estás emparejado con otro participante. Inicialmente, os damos 10€ a cada uno y os invitamos a tomar la misma decisión individual: qué parte de esos 10€ queréis invertir cada uno en una lotería con la posibilidad de obtener un premio de hasta 10€ extra.

Del total de los 10€ puedes invertir lo que quieras en la lotería y la cantidad que no inviertas irá directamente para ti.

El premio lo gana sólo uno de vosotros dos y el ganador se determina en función de lo que cada uno haya invertido en la lotería. El que más invierta tiene mayor probabilidad de conseguir el premio. Éstas son las normas:

- La probabilidad de ganar la lotería es igual al porcentaje que represente tu inversión sobre el total del dinero invertido (entre los dos). Es decir, tu probabilidad de ganar el premio sería igual a $\frac{tuInversion}{tuInversion+suInversion} * 100$. Por tanto, si los dos invertís la misma cantidad, ambos tenéis un 50% de probabilidad de obtener el premio de 10€. Si del total invertido entre los dos, tú has invertido el 60%, entonces tú tendrías un 60% de probabilidad de obtener el premio y el otro participante tendría un 40%. Si él ha invertido el 80% de la suma, entonces, él tiene el 80% y tú solo el 20% de probabilidad de obtener el premio.
- La cantidad que no inviertas irá directamente para ti.
- El dinero que inviertas lo pierdes, independientemente de quién gane el premio.
- Si ninguno de los dos invierte nada, el premio se pierde. Por tanto, cada uno de vosotros se llevaría los 10€ que le damos al principio para invertir pero nada más.
- Si uno de los dos no invierte nada y el otro invierte algo (sea lo que sea), el premio iría para el que sí ha invertido, pues tiene el 100% de la inversión. Por tanto, el que no invirtió nada se llevaría los 10€ que recibió al inicio, mientras que el que sí invirtió se llevaría los 10€ del premio, más los 10€ que recibió para invertir, menos el dinero que invirtió.

Y ahora, sólo para saber que has entendido las normas, por favor responde a estas preguntas:

- Si decides invertir 1€ y ganas el premio, ¿cuánto dinero te llevarías en total?
- ¿Y si inviertes 4€ y no ganas el premio?
- ¿Puedes conseguir el premio si no inviertes nada? (sí o no)
- ¿Puedes conseguir el premio si inviertes menos que el otro? (sí o no)

Ahora tu decisión. De los 10€ que te damos:

¿CUÁNTO INVIERTES EN ESTA LOTERÍA? (múltiplos de 0,50€ entre 0 y 10)

El pago se determinará en función de la decisión anterior. Por tanto, tu decisión ya está tomada. Ahora, para tener más información sobre cómo has tomado tu decisión, por favor, responde a las siguientes preguntas hipotéticas.

Imagina que te informamos de la cantidad que invierte la otra persona. ¿CUÁNTO INVERTIRÍAS EN ESTA LOTERÍA SI SUPIERAS CUÁNTO INVIERTE EL OTRO?:

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

(escribe una cantidad del 0 al 10, en múltiplos de 0,50, en la casilla que hay al lado de cada pregunta)

¿Cuánto invertirías si supieras que el otro invierte 0?

¿Cuánto invertirías si supieras que el otro invierte 1?

...

Version 2 El premio final se determina en función de lo que cada uno haya invertido en la lotería. El que más invierta recibe una parte mayor del premio. Éstas son las normas:

- La parte del premio que recibes es igual al porcentaje que represente tu inversión sobre el total del dinero invertido (entre los dos). Es decir, tu parte del premio sería igual a $\frac{tuInversion}{tuInversion+suInversion} * 100$. Por tanto, si los dos invertís la misma cantidad, ambos os llevaréis el 50% de los 10€ de premio, esto es, 5€ para cada uno. Si del total invertido entre los dos, tú has invertido el 60%, entonces tú te llevarás el 60% del premio (6€) y el otro participante el 40% (4€). Si él ha invertido el 80% de la suma, entonces, él obtiene el 80% (8€) y tú solo el 20% (2€) del premio.
- La cantidad que no inviertas irá directamente para ti.
- El dinero que inviertas lo pierdes, independientemente de qué parte del premio te lleves.
- Si ninguno de los dos invierte nada, el premio se pierde. Por tanto, cada uno de vosotros se llevaría los 10€ que le damos al principio para invertir pero nada más.
- Si uno de los dos no invierte nada y el otro invierte algo (sea lo que sea), el premio iría entero para el que sí ha invertido, pues tiene el 100% de la inversión. Por tanto, el que no invirtió nada se llevaría los 10€ que recibió al inicio, mientras que el que sí invirtió se llevaría los 10€ del premio, más los 10€ que recibió para invertir, menos el dinero que invirtió.

Nota: Ahora irían una serie de preguntas de control, como en la versión 1, así como la elección del participante y otras preguntas de cuánto invertirías si superieras lo que el otro invierte

F.4.4.2. Decisión 2

Estás emparejado con otro participante. Inicialmente, os damos 10€ a cada uno y os invitamos a tomar la misma decisión individual: qué parte de esos 10€ queréis colocar cada uno en un fondo común. Todo lo que no coloques en el fondo es directamente para ti.

Una vez creado el fondo (con las aportaciones de los dos), cada uno recibirá 80 cts. por cada euro que haya en él, venga de quien venga. Es decir, tanto tú como tu pareja recibiréis 0,80 multiplicado por la cantidad total del fondo, independientemente de lo que cada uno haya puesto en él. Ahora pondremos algunos ejemplos.

- Si los dos decidís no colocar nada en el fondo común, los dos acabáis con 10€, los que no habéis puesto en el fondo común, y no os lleváis nada del fondo (porque contiene 0€).
- Si los dos decidís colocar los 10€ en el fondo común, entonces, habría 20€ en su interior. Esto significa que cada uno os llevaríais 16€ ($20 \times 0,80$).
- Si uno de los dos decide no poner nada y el otro decide poner los 10€ en el fondo, entonces, el que no puso nada acaba con 18€ (10 que se quedó más 8 del fondo, $10 \times 0,80$) mientras que su pareja acaba con 8€ (lo que le corresponde del fondo porque no se quedó nada).

Nota: Ahora irían una serie de preguntas de control, como en la versión 1, así como la elección del participante y otras preguntas de cuánto invertirías si superieras lo que el otro invierte

F.4.4.3. Decisión 3

Estás emparejado con otro participante. Inicialmente, os damos 10€ a cada uno y os invitamos a tomar la misma decisión individual: qué parte de esos 10€ queréis colocar cada uno en un fondo común. Todo lo que no coloques en el fondo es directamente para ti.

Una vez creado el fondo (con las aportaciones de los dos), cada uno recibirá 1,20 euros por cada euro que haya en él, venga de quien venga. Es decir, tanto tú como tu pareja recibiréis 1,20 multiplicado por la cantidad total del fondo, independientemente de lo que cada uno haya puesto en él. Ahora pondremos algunos ejemplos.

- Si los dos decidís no colocar nada en el fondo común, los dos acabáis con 10€, los que no habéis puesto en el fondo común, y no os lleváis nada del fondo (porque contiene 0€).

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

- Si los dos decidís colocar los 10€ en el fondo común, entonces, habría 20€ en su interior. Esto significa que cada uno os llevaríais 24€ (20 x 1,20).
- Si uno de los dos decide no poner nada y el otro decide poner los 10€ en el fondo, entonces, el que no puso nada acaba con 22€ (10 que se quedó más 12 del fondo, 10 x 1,20) mientras que su pareja acaba con 12€ (lo que le corresponde del fondo porque no se quedó nada).

Nota: Ahora irían una serie de preguntas de control, como en la versión 1, así como la elección del participante y otras preguntas de cuánto invertirías si superieras lo que el otro invierte

F.4.4.4. Decisiones 4 y 5

Estás emparejado con otro participante. Inicialmente, os damos 20€ para que los dividáis entre la otra persona y tú. Uno de los dos (jugador A) va a proponer cómo dividir los 20€ entre vosotros. El otro (jugador B) puede aceptar o rechazar esa división. Si la rechaza, ninguno de los dos se llevará nada.

Por ejemplo: el jugador A le manda 4€ al jugador B quedándose 16€ para sí mismo, y el jugador B lo acepta. Entonces el jugador A se lleva 16€ y el jugador B, que acepta la división, se lleva 4€. En cambio, si el jugador B no la acepta ninguno de los dos se lleva nada.

Ahora vas a tomar las decisiones de los dos casos, tanto del jugador A como del jugador B (te puede tocar ser tanto A como B).

Decisión 4 Si tú eres el que hace la división (jugador A), ¿qué cantidad le mandas al jugador B? La parte de los 20€ que no le envías es para ti si él acepta tu división. Pero recuerda que si la rechaza ninguno gana nada. Le mandas (en múltiplos de 2€)

Decisión 5 Si tú eres el jugador B puedes aceptar o rechazar la división que te proponga el jugador A. Recuerda que si rechazas ninguno gana nada. Ahora responde si aceptas o rechazas cada una de estas divisiones.

- Si te manda 0€ y se queda 20€ ¿Aceptas o rechazas la división?
- Si te manda 2€ y se queda 18€ ¿Aceptas o rechazas la división?
- Si te manda 4€ y se queda 16€ ¿Aceptas o rechazas la división?

...

- Si te manda 20€ y se queda 0€ ¿Aceptas o rechazas la división?

F.4.4.5. Decisión 6

Estás emparejado con otro participante. Inicialmente, os damos 20€ para que los dividáis entre la otra persona y tú. Uno de los dos (jugador A) va a dividir los 20€ entre vosotros. El otro (jugador B) recibirá la cantidad que el jugador A le envíe pero no tiene que tomar ninguna decisión (el jugador B es pasivo).

Por ejemplo: el jugador A le manda 4€ al jugador B quedándose 16€ para sí mismo. Entonces el jugador A se lleva 16€ y el jugador B se lleva 4€.

Ahora vas a tomar la decisión como jugador A (aunque te puede tocar ser tanto A como B).

Si tú eres el que hace la división (jugador A), ¿qué cantidad le mandas al jugador B? La parte de los 20€ que no le envíes es para ti. Le mandas(en múltiplos de 2€)

F. ¿CÓMO SON NUESTROS VOLUNTARIOS?

Referencias