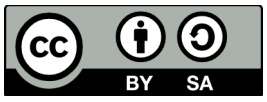
Gestió de màquines de bingo

Administració de web i comerç electrònic en entorns de programari lliure

Autor: **David Vilar Alcalá**
Consultor: **Francisco Javier Noguera Otero**
Data: **24 de gener de 2011**



2. Resum del projecte

Una de les activitats més principals de Ludicus Contest S.L. consisteix en l'explotació de màquines de bingo electrònic a Mèxic. La correcta gestió d'aquestes màquines necessita d'informació immediata sobre el rendiment de cadascun dels jocs instal·lats a cada sala.

En la situació inicial, aquesta informació s'obtenia mitjançant informes mensuals enviats manualment des de Mèxic cap a les oficines de Barcelona, impedint així una resposta àgil a l'evolució dels jocs i produint una minva en els beneficis operatius.

Per millorar aquesta situació, s'ha desenvolupat:

- Un mètode per automatitzar la recol·lecció de dades, aprofitant que les màquines ja disposen de connexió a Internet. Aquest sistema s'ofereix com a serveis web.
- Una aplicació web que permet la consulta i exportació de les dades recollides en els format definit pel client.

Per desenvolupar el sistema s'han escollit solucions de programari lliure. Així, el servidor està codificat amb Ruby on Rails que funcionà sota un servidor web Apache i accedeix a una base de dades PostgreSQL, tot sobre GNU/Linux.

Per a la gestió del desenvolupament s'ha seguit la metodologia àgil Scrum, ja que ofereix una millor adaptació als canvis en els requisits que s'han produït en el desenvolupament del projecte.

Índex

3. Introducció

Ludicus Contest S.L. és una empresa del sector del joc i els casinos amb seu a Barcelona. Tot i que en els seus inicis la seva principal activitat era el desenvolupament de jocs d'apostes *on-line* i de màquines tipus B per Espanya, els seus interessos principals van enfocar-se en el mercat mexicà.

La principal activitat del mercat del recreatiu a Mèxic es desenvolupa al voltant del que s'anomena bingo electrònic, a diferència d'altres mercats on les grans triomfadores són les clàssiques màquines de casino, evolucionades per funcionar amb pantalles de vídeo.

El bingo electrònic deriva del bingo tradicional i el seu joc principal es base en cartrons i una extracció. Tot i així, té força diferències respecte del bingo tradicional que el fan més semblant a una màquina de casino.

Generalment, als bingos electrònics els jugadors poden apostar entre 1 i 4 cartrons, de 5x3 posicions. Cada posició té un número, que correspon amb una bola de l'extracció. A diferència del bingo tradicional, no guanya diners qui primer aconsegueix una determinada combinació (línia i bingo) sinó que existeix una taula de premis més o menys gran. Cada combinació de la taula de premis porta associat un multiplicador que s'aplica a l'aposta actual. A cada partida, s'extreuen 30 boles i es marquen automàticament al cartró del jugador. Al final de la partida, el premi correspon amb els símbols que s'han marcat als cartrons. Per tant, tot i haver extracció comuna, els jugadors no competeixen entre ells per aconseguir el premi.

A aquest joc bàsic se l'afegeix la possibilitat de comprar boles extres. En finalitzar l'extracció, s'ofereix al jugador la compra d'una bola extra, és a dir, una bola de les 30 que queden per extreure. El preu d'aquesta bola extra es calcula en funció dels premis que pot guanyar i de la probabilitat de treure una bola, modificat pel percentatge de pagament desitjat. Aquest joc és molt popular entre els jugadors i sens dubte és una de les claus de l'èxit d'aquest tipus de màquines.

Com en els casinos, les màquines estan interconnectades i ofereixen un premi comú, anomenat progressiu, acumulat o *jackpot*. Aquest premi *jackpot* s'incrementa a cada partida amb un percentatge de l'aposta dels jugadors. Cada joc defineix el seu propi mecanisme d'obtenir-lo, per exemple fent línia amb les 5 primeres boles o bingo abans de la bola 27. La freqüència d'aquest premi és força baixa i, per tant, generalment s'acumulen quantitats molt importants. Aquest també és un factor clau per l'èxit d'aquestes màquines, ja que un *jackpot* elevat és un atractiu molt important pels jugadors.

El model de negoci d'aquestes màquines consisteix en que les sales de bingo cedeixen part del seu espai per a la instal·lació de màquines, generalment a canvi d'un percentatge sobre la recaptació de cadascuna. És l'empresa operadora de les màquines (nosaltres en aquest cas) la responsable del seu manteniment i la seva gestió, és a dir, decidir per cada màquina de cada sala quin joc oferir. Aquesta gestió és un aspecte molt important ja que determinats jocs funcionen millor a determinades sales i, per tant, ofereixen recaptacions més altes i, per tant, major beneficis tant per la sala com per l'operadora. La decisió de quins jocs instal·lar es realitza en funció de les estadístiques de joc i recaptacions a cada sala.

Actualment la gestió es fa mitjançant informes mensuals enviats manualment des de Mèxic per les pròpies sales. Aquesta metodologia de treball s'ha demostrat insuficient, ja que no permet una gestió prou àgil. En una situació ideal hauria de ser possible conèixer diàriament les recaptacions. Així seria possible determinar ràpidament quins jocs funcionen millor en quines sales i, d'aquesta manera, maximitzar els guanys.

Els informes consisteixen en un full de càlcul en un format similar al següent:

Sala	Winpot Los Mochis		**Període**	1/11/09 fins 1/12/09
Màquina	**Crèdit in**	**Crèdit out**	**Win**	**Payout**
1	1000	800	200	80%
...				
N	1000	750	150	75%
Jackpots				
14/11/09	$10.000			

Aquestes dades no aporten suficient informació sobre l'evolució dels jocs a les sales:

- Són de periodicitat mensual.
- Les dades no s'obtenen dels terminals, són els treballadors de les sales qui les proporcionen i són susceptibles a error.
- No ofereixen cap informació sobre els diners jugats en boles extres, punt important en la rendibilitat d'una màquina.

Amb el nou sistema es milloraran aquestes tres mancances.

3.1. Estructura de la memòria

La memòria es troba dividida en els següents apartats:

- Introducció.
- Estudi de viabilitat: es defineixen els objectius i requeriments del projecte, una descripció de la situació actual i les diferents alternatives per assolir-los. S'escollirà la alternativa més adient per al projecte.
- Anàlisi: conté l'especificació detallada de la solució escollida. Per això, es farà una descripció general del sistema i de les seves interfícies d'usuari, a més del pla de proves.
- Disseny: defineix el model arquitectònic del sistema, els diferents subsistemes que el composaran i els casos d'ús disponibles, incloent els diferents actors que hi participaran. També es descriuran les convencions de codificació i l'entorn de desenvolupament que s'ha fer servir.
- Desenvolupament: descriu la metodologia de desenvolupament utilitzada, que en aquest cas ha estat la metodologia àgil anomenada Scrum. Es mostra la organització del codi font i com s'han realitzat les proves per tal de garantir la correctesa de la solució.
- Implantació: explica el procés d'implantació i la seva relació amb la fase de desenvolupament.
- Conclusions: consisteix en un resum dels objectius assolits i les possibles ampliacions d'aquest projecte.

4. Estudi de viabilitat

4.1. Objectius del projecte

L'objectiu principal d'aquest projecte és millorar la gestió del parc de màquines de bingo per tal de maximitzar els beneficis, disposant de dades més freqüents i actualitzades sobre el seu funcionament.

Per aconseguir-ho, caldrà fer el següent:

- Disposar de dades actualitzades de les màquines de joc, com per exemple el tipus de joc que tenen instal·lat.
- Disposar de dades històriques de les màquines de joc, que consistiran en la recaptació diària.
- Disposar de resums de les recaptacions per joc i sala en un període de temps.
- Disposar de llistats dels darrers *jackpots* a cada sala.

Per tal d'assolir els objectius caldrà millorar l'obtenció de dades per aconseguir que sigui automàtica i freqüent. S'aprofitarà que les màquines ja disposen de connexió a Internet per tal que siguin elles mateixes les que insereixin les dades en el sistema de forma automàtica.

També caldrà oferir alguna aplicació per tal de permetre als responsables de la pressa de decisions accedir a aquestes dades de forma senzilla i sense necessitat de fer-ho directament contra la base de dades.

Així, els productes resultants d'aquest projecte seran dos:

- Un conjunt de *web services* accessibles per les màquines mitjançant Internet i que permetin introduir les dades necessàries, consistents en resum de jugades i *jackpots*.
- Una aplicació web per tal de poder consultar i filtrar la informació requerida per a la pressa de decisions. Això és així donat que l'accés a la informació per part dels responsables ha d'estar disponible des de qualsevol punt, ja que aquesta es consultarà tant des de Barcelona, Itàlia i Mèxic.

Caldrà a més tenir en compte que el tractament avançat d'aquestes dades es farà mitjançant programari extern, com fulls de càlcul. Serà així necessari disposar d'aquestes dades en algun format fàcilment importable per part d'aquest programari, com per exemple en format *.csv*[1].

4.2. Definició de requisits del sistema

A partir dels objectius del projecte, el producte resultant haurà de satisfer els següents requisits:

- Les dades han d'estar disponibles tant en format web com en format .csv per tal de permetre el seu tractament amb aplicacions externes com fulls de càlcul.
- Un cop en producció serà recomanable utilitzar servidors existents, aprofitant que aquests disposen de recursos disponibles. El maquinari actual disposa del següent programari: Debian GNU/Linux com a sistema operatiu, PostgreSQL com a base de dades i Apache com a servidor web. Així aconseguirem els següents beneficis:
  - El nou sistema estarà ja integrat en les polítiques de seguretat i de còpies de seguretat ja existents.
  - Eliminem els costos de nou maquinari.
- Ha de ser possible accedir a les dades emmagatzemades a la base de dades des de qualsevol lloc sense necessitat de programari addicional. Per això, es farà servir una aplicació web que compleixi l'estàndard W3C HTML 4.01. Especialment haurà de garantir-se el seu correcte funcionament en el navegador Mozilla Firefox en la seva versió més recent (actualment és la 3.6.3) ja que és l'utilitzat en els ordinadors dels treballadors.
- La interfície ha de ser senzilla i atractiva, donant prioritat a la usabilitat i la velocitat d'ús.

1 *Comma-separated values*

- Els terminals han de ser capaços d'enviar les dades de forma automàtica al sistema. Cal tenir especial cura en garantir l'existència de llibreries per Java que permetin la comunicació, ja que els programari dels terminals està desenvolupat amb aquest llenguatge.
- Les necessitats de configuració de xarxa dels clients per comunicar-se amb l'API han de ser mínimes, ja que generalment la connexió amb Internet a les sales és administrada per la pròpia sala.
- La comunicació haurà de ser autentificada, de tal forma que només les persones o aplicacions autoritzades puguin accedir al sistema. També haurà de ser xifrada, ja que les dades que es comunicaran són confidencials.
- Ha de fer servir únicament programari lliure i estàndards oberts.
- Ha de ser extensible a futurs desenvolupaments de l'empresa. Futurs productes tindran problemàtiques similars i s'hauria de poder utilitzar també per aquests.

4.3. Estudi de les alternatives i solució triada

No existeix cap aplicació disponible, ja sigui programari lliure o propietari, que satisfaci els requisits del projecte. És gairebé segur que la major part d'empreses que operen amb màquines disposen d'alguna eina que ofereixi unes funcionalitats similars a les requerides pel nostre projecte, però consisteixen en programari d'ús intern i en cap cas es fan públiques. Per tant, serà necessari implementar la solució en gran part.

Respecte al sistema operatiu del servidor, el sistema gestor de bases de dades i el servidor web, un dels requisits ens recomana fer servir GNU/Linux, PostgreSQL i Apache. Els tres components són completament apropiats als nostres requisits.

GNU/Linux és un sistema operatiu de programari lliure, distribuït sota llicència GNU GPL i d'altres. És un sistema operatiu derivat de UNIX, iniciat l'any 1991 per Linus Torvalds, qui va escriure la primera versió del nucli anomenat Linux. El projecte GNU, iniciat l'any 1983 pel Richard Stallman, és el responsable de la major part de les eines i llibreries d'usuari, com el compilador GCC. La unió d'ambdós elements formen el sistema operatiu.

És un sistema operatiu de propòsit general i amplia difusió, amb un gran suport tant de la comunitat com d'un gran nombre empreses, com IBM, Intel o Google.

PostgreSQL és una base de dades relacional distribuïda sota la llicència lliure BSD. Es troba disponible per diferents sistemes operatius, incloent GNU/Linux. Ofereix suport de l'estàndard SQL i és considerat una alternativa d'altes capacitats i rendiment. Té una forta comunitat al darrera, que ofereix abundant documentació i suport.

El servidor web Apache és el principal projecte de la Apache Software Foundation. És distribueix sota la llicència lliure Apache 2.0 i es troba disponible per diferents sistemes operatius, un d'ells GNU/Linux. És el servidor web més utilitzat i de major èxit: al novembre del 2010, gairebé el 60% de les webs oferides provenen de servidors Apache[1].

La interfície amb la API es farà mitjançant *web services* per tal d'aconseguir independència de la plataforma i garantir la interoperabilitat amb diferents clients.

Un *web service* és un sistema de programari creat per oferir interacció entre màquines remotes mitjançant una xarxa de comunicació. Aquests serveis generalment són accessibles mitjançant l'intercanvi de missatges XML sota el protocol HTTP[2]. En el nostre cas implementarem els serveis web necessaris seguint l'arquitectura REST[3].

L'arquitectura REST fa servir les operacions del protocol HTTP, és a dir GET, POST, PUT i DELETE, per oferir l'accés remot als recursos. És una arquitectura orientada als recursos i no als missatges o a les operacions.

Respecte al *framework* d'aplicacions web, existeixen diverses alternatives de programari lliure. L'escollida és Ruby on Rails[4], que pemet el desenvolupament d'aplicacions web seguint l'arquitectura model-vista-controlador, de forma ràpida i amb un codi mínim, senzill i de qualitat[5].

Ruby on Rails ofereix suport per als principals protocols i arquitectures per oferir *web services*, entre ells REST des de la versió 1.2[6].

El projecte té al darrera una gran comunitat i molta documentació i de qualitat. Es troba disponible per un gran nombre de sistemes operatius, entre ells GNU/Linux, i ofereix suport tant per al servidor de pàgines web Apache com per a la base de dades PostgreSQL.

Ruby on Rails permet el desenvolupament de planes web mitjançant les llibreries Prototype[7] i script.aculo.us[8], que farem servir per al desenvolupament del client web.

Prototype és un framework escrit en JavaScript creat per Sam Stephenson l'any 2005 per facilitar el desenvolupament d'aplicacions Ajax[9] sota Ruby on Rails. És programari lliure i es distribueix sota la lliència MIT. Es troba implementat en un únic fitxer, anomenat prototype.js, que facilita l'accés a l'objecte XMLHttpRequest[10] del navegador.

La llibreria script.aculo.us ofereix efectes visuals dinàmics per aplicacions Ajax. Es distribueix sota la llicència de programari lliure MIT.

5. Anàlisi

5.1. Definició del sistema

De l'anàlisi dels requisits del sistema he definit un seguit de casos d'ús que defineixen les diferents funcionalitats que oferira aquest, els actors que hi podran accedir i les relacions entre els diferents casos d'ús.

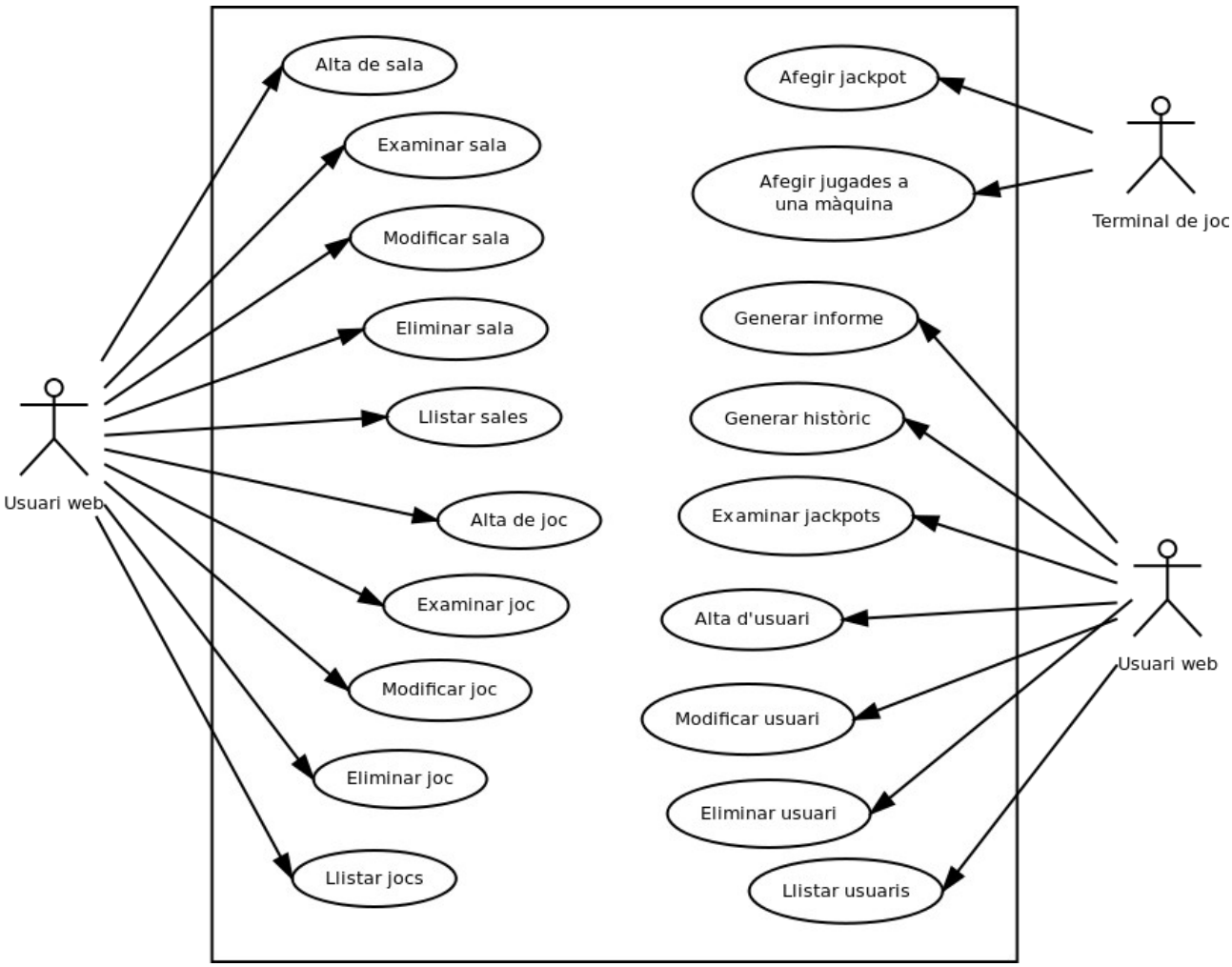
5.1.1. Definició dels actors

Existeixen dos actors diferents que interactuaran amb el sistema:

Usuari web	Té la responsabilitat de crear i gestionar els diferents elements que componen el sistema: sales, màquines i jocs. També podran afegir incidències. Serà aquest actor qui s'encarregarà d'obtenir els informes sobre recaptacions. Interactuaran amb el sistema mitjançant un navegador web.

Terminal de joc	Correspon a les màquines de bingo de les sales. Seran responsables d'afegir periòdicament les jugades que s'hi han realitzat. A més, també podran afegir incidències. Es comunicaran amb el sistema mitjançant els *web services* oferts.

5.1.2. Diagrama de casos d'ús



5.1.3. Definició dels casos d'ús

Nom	**Alta de sala**
Objectius	Donar d'alta una sala amb els seus valors inicials Una sala té els següents atributs: nom, horari, zona horària i comentaris.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona inicia l'alta de la sala. 2. Es presenta un formulari web on el client ha d'introduir les dades. 3. El client introdueix les dades característiques de la sala i les dades inicials de les màquines. 4. El sistema crea la nova sala a la base de dades.
Alternatives	Cap.

Nom	**Modificar sala**
Objectius	Modificar qualsevol de les dades que defineixen una sala.
Actors	Usuari web
Precondicions	La sala existeix.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari escolleix una sala per examinar. 2. El sistema mostra les dades de la sala. 3. L'usuari indica que vol modificar la sala. 4. El sistema mostra la pantalla de modificació de la sala. 5. L'usuari realitza els canvis oportuns i accepta els canvis. 6. El sistema modifica els valors a la base de dades.
Alternatives	Cap.

Nom	**Eliminar sala**
Objectius	Eliminar qualsevol de les dades que defineixen una sala. Encara que la sala tingui jugades associades, aquestes no s'esborraran.
Actors	Usuari web
Precondicions	La sala existeix.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari escolleix una sala per examinar. 2. El sistema mostra les dades de la sala. 3. L'usuari escolleix eliminar la sala. 4. El sistema demana confirmació. 5. L'usuari confirma que vol eliminar la sala. 6. El sistema elimina la sala de la base de dades.
Alternatives	Cap.

Nom	**Llistar sales**
Objectius	Mostrar un llistat de les sales que estan donades d'alta al sistema. Al llistat han d'aparèixer les següents dades: nom de la sala, data de la última partida jugada a qualsevol de les seves màquines i data i quantita de l'últim jackpot guanyat a la sala.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona la opció de llistar les sales. 2. El sistema mostra el llistat de sales.
Alternatives	Cap.

Nom	**Examinar sala**
Objectius	Mostrar a l'usuari un resum de les dades d'una sala. Aquest resum inclou el següent: • Nom, horari, zona horària de la sala i comentaris. • Data de la primera i última partides jugades a qualsevol de les màquines associades. • Llistat de màquines, que inclouen: ◦ Identificador dins la sala. ◦ Data i joc de la primera i última partides jugades a la màquina. ◦ Percentatge real de pagament de tota la història. ◦ Percentatge real de pagament de l'últim mes. • Data, quantitat i joc dels últims 5 jackpots guanyats a la sala.
Actors	Usuari web
Precondicions	La sala existeix.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona la sala de la qual vol veure el resum. 2. El sistema mostra les dades a l'usuari.
Alternatives	Cap.

Nom	**Alta de joc**
Objectius	Crear un nou joc. Un joc es defineix per les dades següents: identificador, nom, imatge i comentaris. La imatge del joc és opcional.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari escolleix donar d'alta un nou joc i introdueix els valors que el defineixen. 2. El sistema dona d'alta el nou joc.
Alternatives	Cap.

Nom	**Examinar un joc**
Objectius	Permet a l'usuari consultar informació sobre un joc. Les dades a mostrar són les següents: • Data de la primera i última jugades corresponents al joc. • Número total de partides jugades. • Payout real del joc. • Payout real del joc, sense comptar boles extres. • Payout de la boles extres.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari escolleix el joc a consultar del llistat de jocs. 2. El sistema mostra les dades del joc.
Alternatives	Cap.

Nom	**Modificar joc**
Objectius	Modificar les dades d'un joc.
Actors	Usuari web
Precondicions	El joc existeix.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona un joc per ser examinat. 2. El sistema mostra les dades del joc. 3. L'usuari escolleix modificar les dades del joc i introdueix les noves. 4. El sistema realitza els canvis a la base de dades.
Alternatives	Cap.

Nom	**Eliminar un joc**
Objectius	Eliminar un joc de la base de dades. Encara que el joc tingui jugades associades, aquestes no s'esborraran.
Actors	Usuari web
Precondicions	El joc existeix.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari escolleix un joc. 2. El sistema mostra les dades del joc. 3. L'usuari escolleix eliminar el joc. 4. El sistema elimina el joc de la base de dades.
Alternatives	Cap.

Nom	**Llistar jocs**
Objectius	Mostrar un llistat dels jocs existents. A la llista ha d'aparèixer la imatge del joc, o el seu nom si no en té imatge definida.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona la opció de llistar els jocs. 2. El sistema mostra el llistat de jocs.
Alternatives	Cap.

Nom	**Afegir jugades a una màquina**
Objectius	Introduir dades sobre la recaptació d'una màquina. Aquestes dades consisteixen en un resum de les partides jugades a la màquina. Cada element té els següents valors: identificador de sala, identificador de màquina, identificador de joc, data i hora, diners totals apostats, diners total guanyats, diners apostats en bola extra, diners guanyats en bola extra, diners guanyats en jackpots.
Actors	Terminal de joc
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. El terminal de joc envia les jugades realitzades a la màquina. 2. El sistema insereix les dades in envia una confirmació.
Alternatives	Si en el moment d'inserir les jugades la màquina no es troba a la base de dades, s'ha d'informar al responsable. Si en el moment d'inserir les jugades el joc no es troba a la base de dades, s'ha d'informar al responsable.

Nom	**Generar informe**
Objectius	Obtenir un resum de les dades de recaptació. Les dades s'han de poder agrupar per joc i sala. Les dades s'han de poder filtrar per sala, joc i data. Les dades es mostraran tant com una pàgina web exportarbles al format *.csv*.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona els filtres a aplicar. 2. El sistema presenta les dades, que consisteixen en el següent: • credit in • credit out • win • payout real • credit in en bola extra • credit out en bola extra • número de partides • número de màquines
Alternatives	Cap.

Nom	**Generar històric**
Objectius	Obtenir un històric de les dades de recaptació, agrupats per setmanes. Les dades s'han de poder agrupar per joc i sala. Les dades s'han de poder filtrar per sala, joc i data. Les dades es mostraran com una pàgina web, però també serà possible exportar-les en format *.csv*.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona els filtres a aplicar. 2. El sistema presenta les dades agrupades per setmanes, que consisteixen en el següent: • credit in • credit out • win • payout real • credit in en bola extra • credit out en bola extra • número de partides
Alternatives	Cap.

Nom	**Generar jackpots**
Objectius	Obtenir un llistat dels últims jackpots. Les dades s'han de poder filtrar per sala i data. Es pot escollir el número de jackpots a mostrar.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari selecciona els filtres a aplicar. 2. El sistema presenta les dades, que consisteixen en el següent: • sala • màquina • joc • data • quantitat
Alternatives	Cap.

Nom	**Alta d'usuari**
Objectius	Crear un usuari autoritzat. Les dades d'un usuari són nom i paraula clau.
Actors	Usuari web
Precondicions	No existeix cap usuari amb aquest nom. La paraula clau ha de tenir almenys 5 caràcters.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari envia les dades del nou usuari mitjançant l'aplicació web. 2. El sistema dóna d'alta el nou usuari.
Alternatives	Cap.

Nom	**Modificar usuari**
Objectius	Modifica la paraula clau d'un usuari existent.
Actors	Usuari web
Precondicions	L'usuari existeix. La nova paraula clau ha de tenir almenys 5 caràcters.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari envia el nom de l'usuari a modificar i la nova paraula clau mitjançant l'aplicació web. 2. El sistema modifica la paraula clau l'usuari.
Alternatives	Cap.

Nom	**Eliminar usuari**
Objectius	Elimina un usuari existent.
Actors	Usuari web
Precondicions	L'usuari existeix. L'usuari no és el mateix amb el qual s'ha identificat.
Usa	Llistar usuaris.
Flux bàsic d'esdeveniments	1. L'usuari envia el nom de l'usuari a eliminar mitjançant l'aplicació web. 2. El sistema elimina l'usuari.
Alternatives	Cap.

Nom	**Llistar usuaris**
Objectius	Mostra una llista dels usuaris existents.
Actors	Usuari web
Precondicions	Cap.
Usa	Cap.
Flux bàsic d'esdeveniments	1. L'usuari envia la petició de veure el llistat d'usuaris. 2. El sistema mostra el llistat d'usuaris.
Alternatives	Cap.

5.2. Definició d'interfícies d'usuari

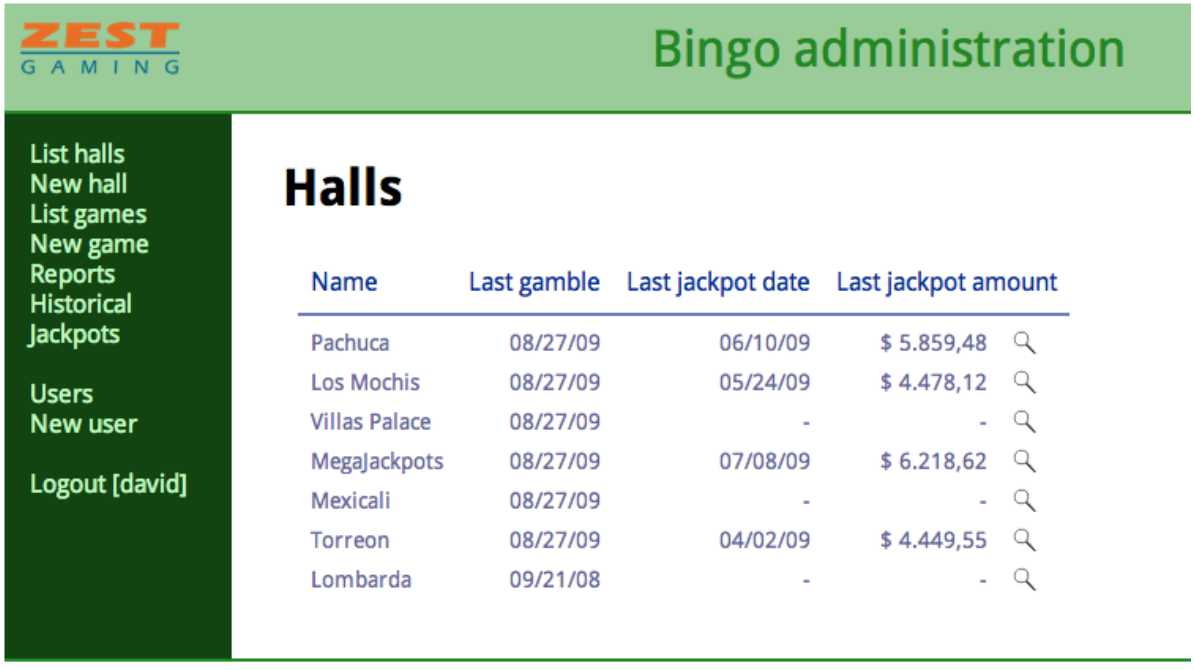
5.2.1. Perfils d'usuari

Tindrem un únic perfil d'usuari, de tipus no tècnic tot i habituat al treball amb ordinadors. En especial, serà competent en la utilització de programes d'edició de documents i fulls de càlcul. També se'l suposa un coneixement suficient de la navegació mitjançant pàgines web.

5.2.2. Principis generals de la interfície de usuari

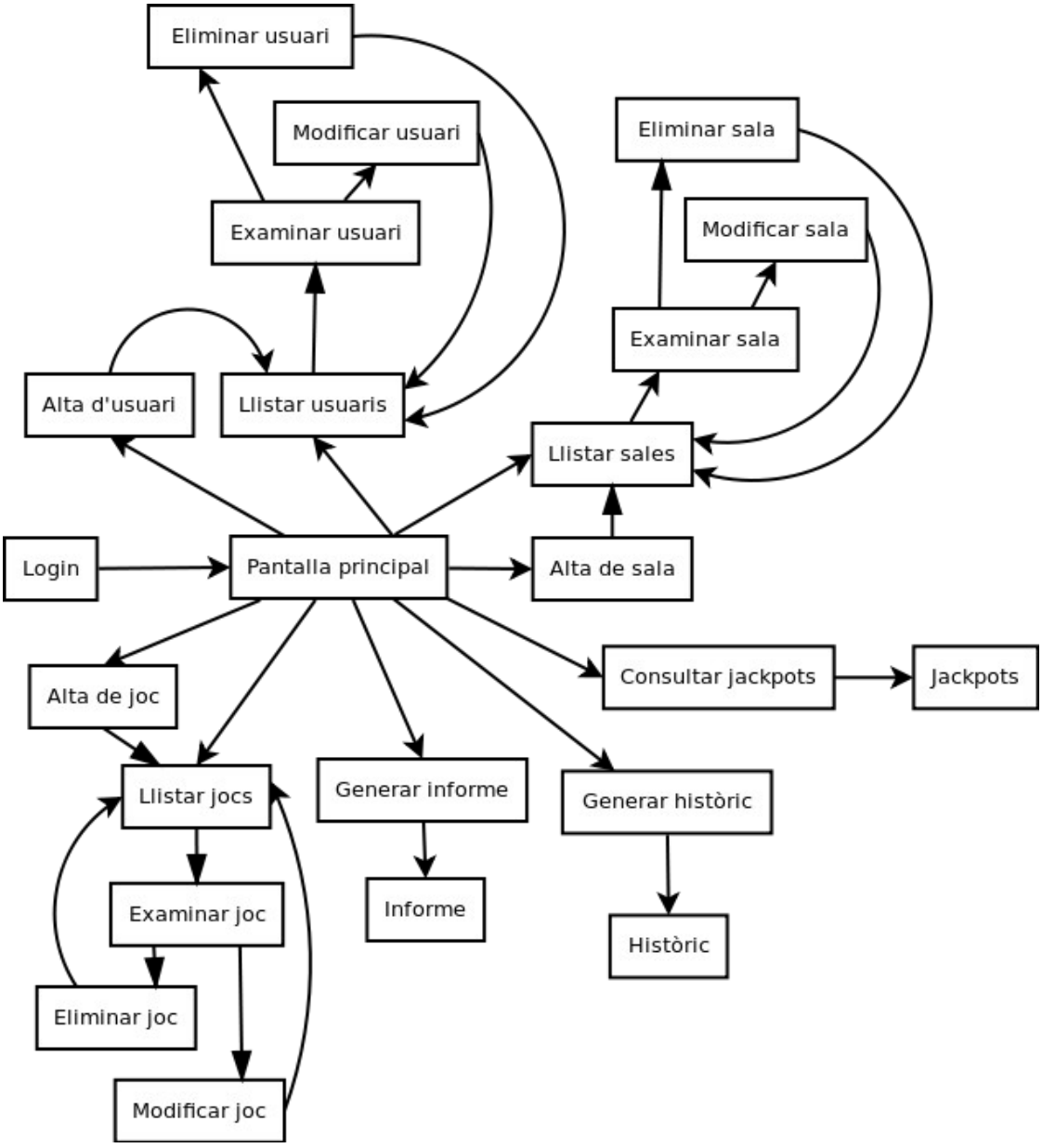
L'aplicació ha de tenir les següents característiques:

- L'accés es realitzarà mitjançant un navegador web.
- S'intentarà que la interfície sigui àgil, reduint en la mesura del possible les transicions entre diferents pàgines web i limitant el número de clicks necessaris per realitzar les opcions més comunes.
- A totes les pàgines apareixeran sempre dos elements:
  - El menú de l'aplicació a l'esquerra, que oferirà els següents enllaços: Llistar sales, Afegir sala, Llistar jocs, Afegir joc, Generar informe, Generar històric, Llistar usuaris, Nou usuari, Sortir del sistema.
  - El panell principal a la part dreta, ocupant la major part de la pantalla, on es mostraran els formularis i les dades de l'aplicació.
- Els missatges d'error o d'avís seran mostrats per pantalla, mitjançant un missatge suficientment clar.
- En cas d'error o avís intern que no es pugui mostrar a l'usuari, s'enviarà aquest per correu electrònic a l'administrador.
- No caldrà integrar una ajuda en la mateixa aplicació.



5.2.3. Diagrama de flux de l'aplicació

La transició entre les diferents pantalles de l'aplicació web seguirà el següent flux.



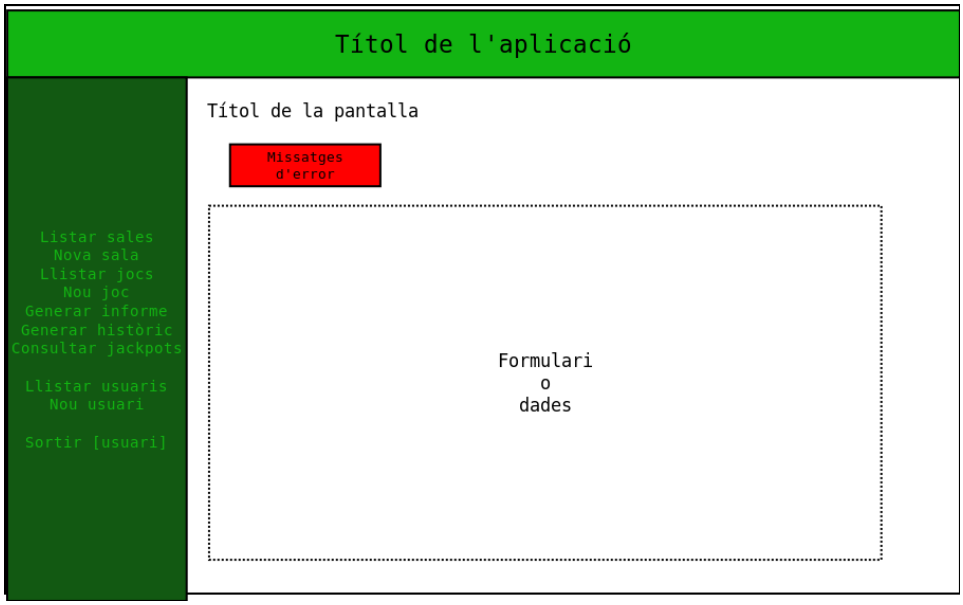
5.2.4. Disseny d'interfícies d'usuari

La pantalla inicial de l'aplicació servirà per tal de permetre a l'usuari autenticar-se en el sistema. La seva disposició serà molt senzilla, mostrant només els camps de nom d'usuari i contrasenya i un botó per accedir-hi.

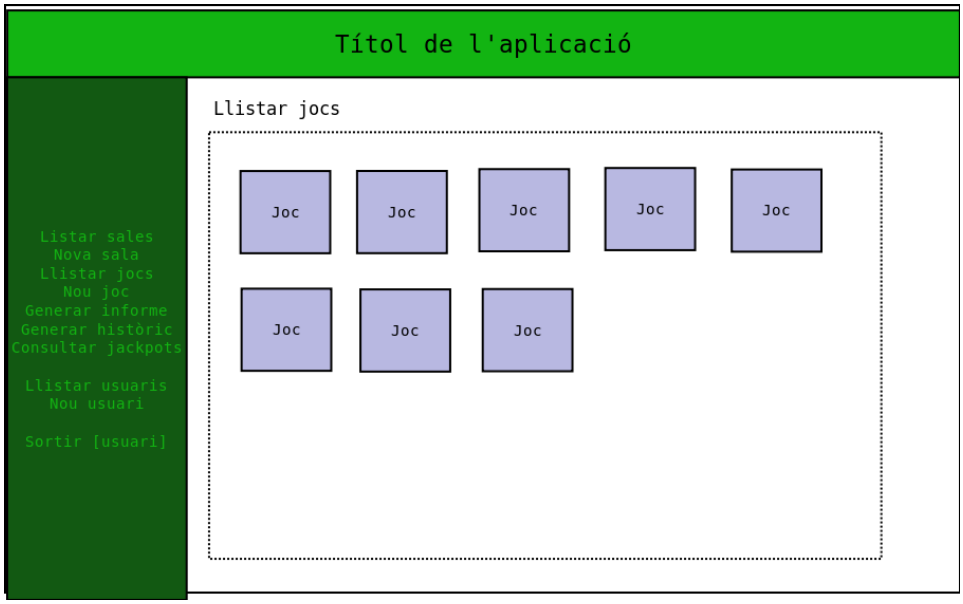
La resta de pantalles tindran la mateixa disposició bàsica. A l'esquerra podrem trobar el menú principal, que serà accessible des de qualsevol pantalla de l'aplicació i permetrà accedir a les següents opcions: Llistar sales, Nova sala, Llistar jocs, Nou joc, Generar informe, Generar històric, Consultar jackpots, Llistar usuaris, Nou usuari, Sortir. Aquesta darrera opció mostrarà, a més, el nom de l'usuari amb el qual s'ha identificat.

Els missatges d'error només apareixeran si s'han produït. En cas contrari, no apareixerà res.

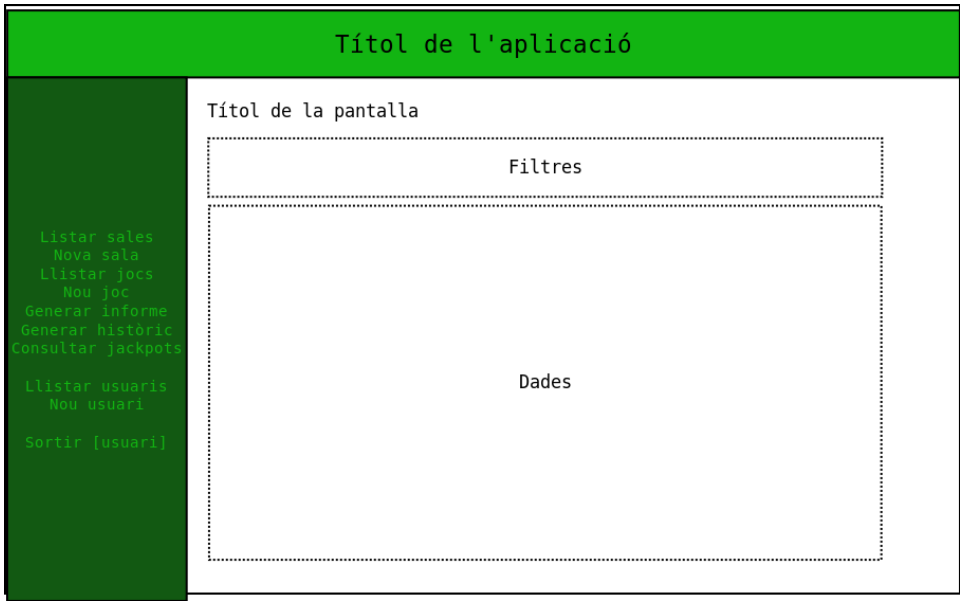
A l'àrea de treball es mostrarà la interfície particular de cadascuna de les pantalles de l'aplicació, ja sigui un formulari, o un llistat.



La pantalla de llistar jocs només mostrarà els jocs disponibles en el sistema, per tal que puguin ser seleccionats per l'usuari. Si tenen imatge assignada, es mostrarà aquesta. En cas contrari, apareixerà el nom del joc.



Les tres opcions que ens mostren informes (Generar informe, Generar històric i Consultar jackpots) tindran una interfície similar. En aquest cas, a la part superior es mostrarà un formulari per tal de seleccionar les característiques de l'informe. En la part inferior, es mostraran les dades de l'últim informe generat.



5.3. Especificació del pla de proves

5.3.1. Proves unitàries

Tal com suggereixen les metodologies de desenvolupament de programari més recents, el desenvolupament es realitzarà orientat a les proves. D'aquesta forma, durant la fase de codificació el primer pas serà definir les proves unitàries que haurà de passar el codi per tal que es consideri correcte. Així podrem garantir una millor qualitat del codi generat.

La pròpia API de Ruby on Rails ofereix suport per a l'execució de proves unitàries[2], fins i tot simulant peticions d'un navegador i, de fet, ofereix una base ideal per tal de desenvolupar mitjançant aquestes metodologies.

El sistema defineix dos tipus de proves automàtiques segons l'element de l'arquitectura MVC al qual es refereixen:

- ***Unit tests***, per provar els models.
- ***Functional tests***, per provar els controladors.

L'execució de totes aquestes proves es pot automatitzar mitjançant l'eina Rake[3] que es troba integrada en molts IDEs de Ruby on Rails. D'aquesta forma el procés de proves queda perfectament lligat amb la codificació.

A banda d'aquests tipus de test, Ruby on Rails permet definir ***Integration tests*** per tal de provar les interaccions entre diferents controladors. Gràcies a aquestes proves podrem garantir el correcte funcionament dels mètodes oferts mitjançant l'API remota.

Per exemple, el següent algorisme correspon a una prova d'integració per comprovar la inserció de jugades.

1. Es fa *login* en el sistema
2. Es consulten les dades d'una sala que no existeix. S'ha de produir un error.
3. S'insereix aquesta sala.
4. Es demana un llistat de les sales. Ha de contenir la nova sala.
5. Es consulten les dades d'aquesta sala. El sistema ha de retornar les seves dades.
6. S'envia una petició per esborrar la sala. El resultat ha de ser correcte.
7. Es demana un llistat de les sales. No ha de contenir la sala.
8. Es consulten les dades de la sala esborrada. S'ha de produir un error.
9. Es torna a inserir la sala. El resultat ha de ser correcte.
10. S'insereixen jugades de la sala corresponents al mes de maig. El resultat ha de ser correcte.
11. S'envia una petició per esborrar la sala. S'ha de produir un error.
12. Es demana un informe de la sala corresponent al mes de maig. Ha de contenir les dades introduïdes.
13. S'envien noves jugades de la sala, corresponents al mes de juny. El resultat ha de ser correcte.
14. Es demana un informe de la sala corresponent al mes de maig. Ha de reflectir només les dades de la primera introducció.
15. Es demana un informe de la sala corresponent al mes de juny. Ha de reflectir només les dades de la segona introducció.
16. Es demana un informe de la sala corresponent al mes de juliol. Ha de ser buit.
17. Es demana un informe de la sala corresponent als mesos de maig i juny. Ha de contenir les dades d'ambdues introduccions.
17. S'insereixen per duplicat les dades del mes de maig. El sistema ha d'ignorar-ho i retornar correctament.
18. Es demana un informe de la sala corresponent al mes de maig. Ha de reflectir només les dades de la primera introducció, ignorant les dades duplicades.
19. Se surt del sistema.

2 Per més informació, es pot consultar el document A Guide to Testing Rails Applications a la pàgina de Ruby on Rails.
3 Rake és l'equivalent de l'eina *make* per Ruby.

Un altre exemple, relatiu als jocs.

1. Es fa *login* en el sistema
2. Es dóna d'alta un joc. El resultat ha de ser correcte.
3. Es consulten les dades d'aquest joc. Les dades han de coincidir amb les introduïdes.
4. Es modifiquen les dades del joc. El resultat ha de ser correcte.
5. Es consulten les dades d'aquest joc. Les dades han de coincidir amb les modificades.
6. S'elimina el joc. El resultat ha de ser correcte.
7. Es dóna d'alta un nou joc. El resultat ha de ser correcte.
8. S'insereixen jugades corresponents a aquest nou joc.
9. Es modifiquen les dades del joc. El sistema ha de mostrar un avís i demanar confirmació
10. Es confirma i el resultat és correcte.
11. Es consulten les dades d'aquest joc. Les dades han de coincidir amb les modificades.
12. S'elimina el joc. El sistema ha de mostrar un avís i demanar confirmació.
13. Es confirma i el resultat és correcte.
14. Es consulten les dades d'aquest joc. El sistema ha de retornar error perquè aquest no existeix.
15. Es consulten les jugades introduïdes corresponents al joc. Aquestes han de continuar existint.
16. Se surt del sistema.

Aquestes proves es codificaran amb Ruby fent servir l'API corresponent de Ruby on Rails, a mesura que es vagin afegint funcions al sistema.

5.3.2. Proves d'acceptació

A més de les proves unitàries caldrà definir una sèrie de proves d'acceptació per tal que el client pugui donar el vist i plau a l'aplicació. Aquestes proves s'aniran realitzant a mesura que es vagin afegint característiques durant el desenvolupament.

Les proves d'acceptació se centraran en validar la interfície de l'usuari amb el sistema, és a dir, les interaccions per part de l'usuari amb l'aplicació web. En el nostre cas les proves d'acceptació les realitzarà algun dels que finalment seran els seus usuaris.

El guió de les proves d'acceptació és el següent:

Prova d'acceptació	
Provador	**Data**

Descripció	El login i logout en l'aplicació web funcionen correctament. No es pot accedir a cap opció de l'aplicació sense haver fet login.	**Resultat**
Comentaris		

Descripció	La navegació per les diferents opcions de l'aplicació és prou intuïtiva i usable i es correspon amb el flux de l'aplicació definit.	**Resultat**
Comentaris		

Descripció	Els llistats de recaptació són clars i mostren la informació dessitjada. Es possible definir un llistat seleccionant sales, jocs i dades d'inici i final. Es pot aconseguir una versió del llistat en format *.csv*. És possible importar les dades obtingudes a una aplicació de fulls de càlcul.	**Resultat**
Comentaris		

Descripció	Els llistats de sales són clars i permeten trobar una sala ràpidament. La consulta de dades d'una sala és prou clara. Es poden modificar les dades d'una sala de forma senzilla	**Resultat**
Comentaris		

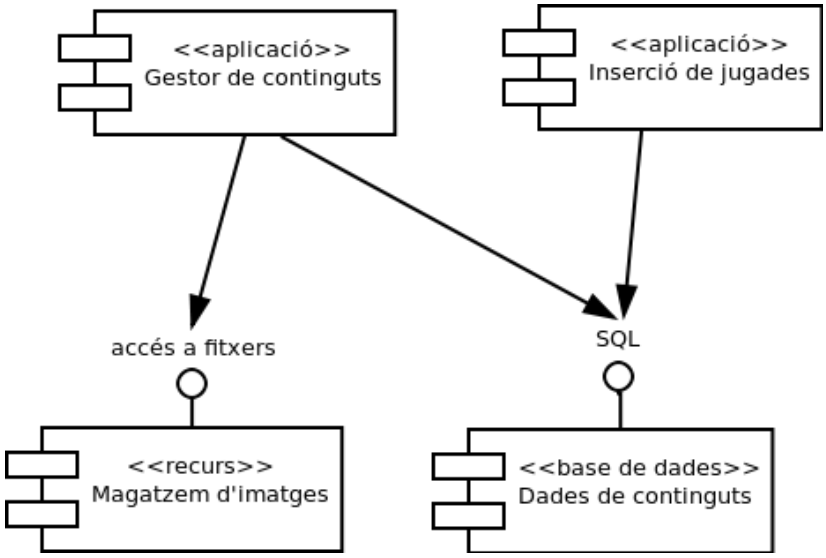
Descripció	Es pot obtenir un llistat dels jocs existents en el sistema. L'alta, modificació i eliminació de jocs és correcte. Si s'intenta modificar o eliminar un joc amb partides introduïdes es produeix un avís. Aquest avís és prou clar i entenedor.	**Resultat**
Comentaris		

Aquestes proves s'hauran de començar a realitzar tan bon punt sigui possible per tal de corregir els possibles errors o deficiències en el disseny com més aviat millor.

6. Disseny

6.1. Arquitectura del sistema

6.1.1. Diagrama UML de components



6.1.2. Definició de components

Gestor de continguts	
Permet gestionar sales, màquines, jocs i usuaris. Mostra les dades emmagatzemades al sistema (sales, jocs, màquines i jugades). Permet controlar l'accés a les dades només als usuaris autoritzats.	Dades de continguts. Magatzem d'imatges.

Dades de continguts	
Emmagatzema les dades inserides.	Gestor de continguts.

Magatzem d'imatges	
Emmagatzema al sistema de fitxers les imatges dels jocs.	Gestor de continguts.

Inserció de jugades	
Ofereix un servei web que permet afegir jugades a les màquines de les sales. Ofereix un servei web que permet afegir *jackpots* a les màquines de les sales. Permet l'accés al sistema només a usuaris autoritzats.	Dades de continguts.

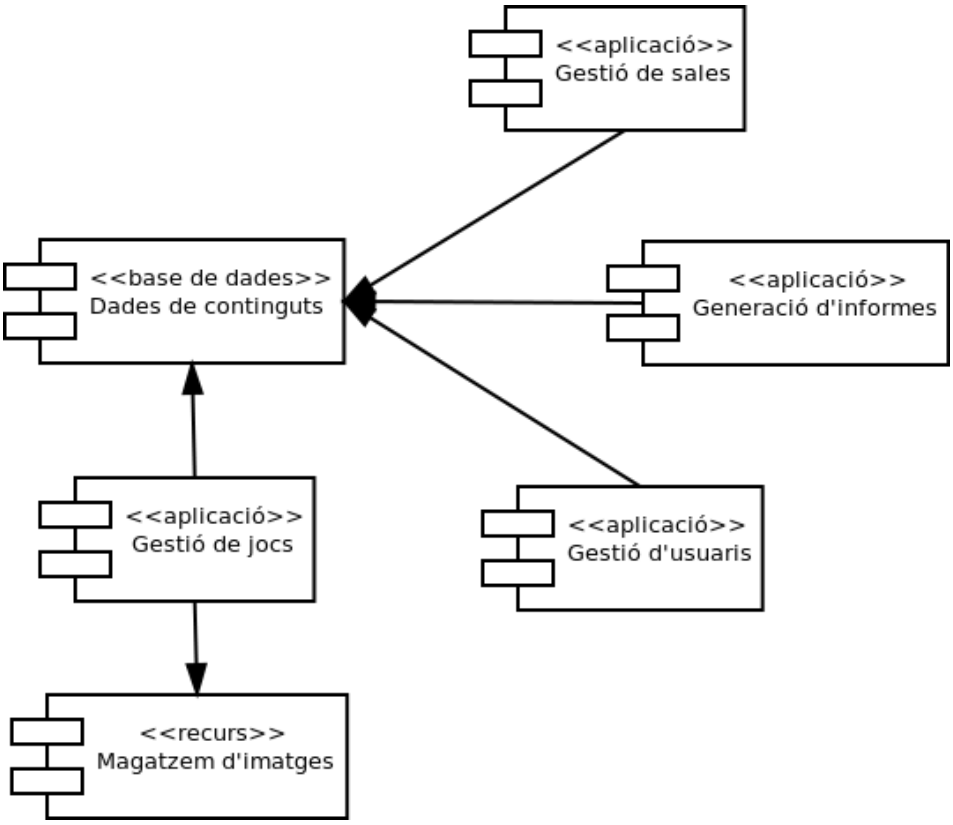
6.2. Descripció de subsistemes

Únicament és necessari subdividir el component Gestor de continguts. Els altres tres components estan suficientment definits i no requereixen de cap subdivisió.

6.2.1. Subsistemes del component Gestor de continguts

El component Gestor de continguts estarà compost pels subsistemes següents:

- Subsistema de gestió de sales (permetrà gestionar sales i les màquines associades).
- Subsistema de gestió de jocs.
- Subsistema de gestió d'usuaris (permetrà gestionar els usuaris i controlar l'accés només als autoritzats).
- Subsistema d'informes (permetrà accedir als diferents informes i generar fitxers *.csv*).



6.3. Revisió de casos d'ús

6.3.1. Revisió dels subsistemes segons el cas d'ús

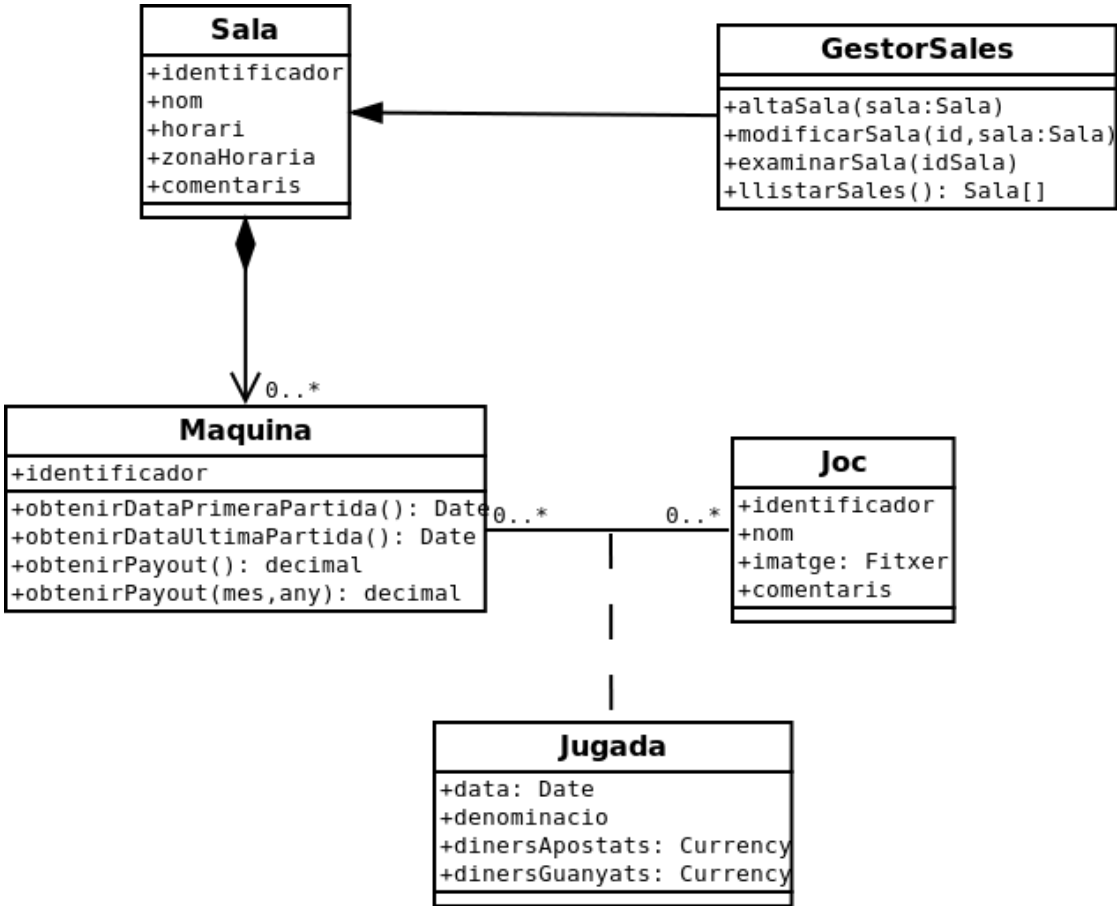
Relatius a sales i màquines

Tots els casos d'ús relatius a sales i màquines, és a dir:

- Alta de sala, Llistar sales, Examinar sala, Modificar sala, Eliminar sala

necessitaran de la intervenció del subsistemes **Gestió de sales** i **Dades de continguts**.

La comunicació entre els subsistemes es realitzarà mitjançant sentències SQL.

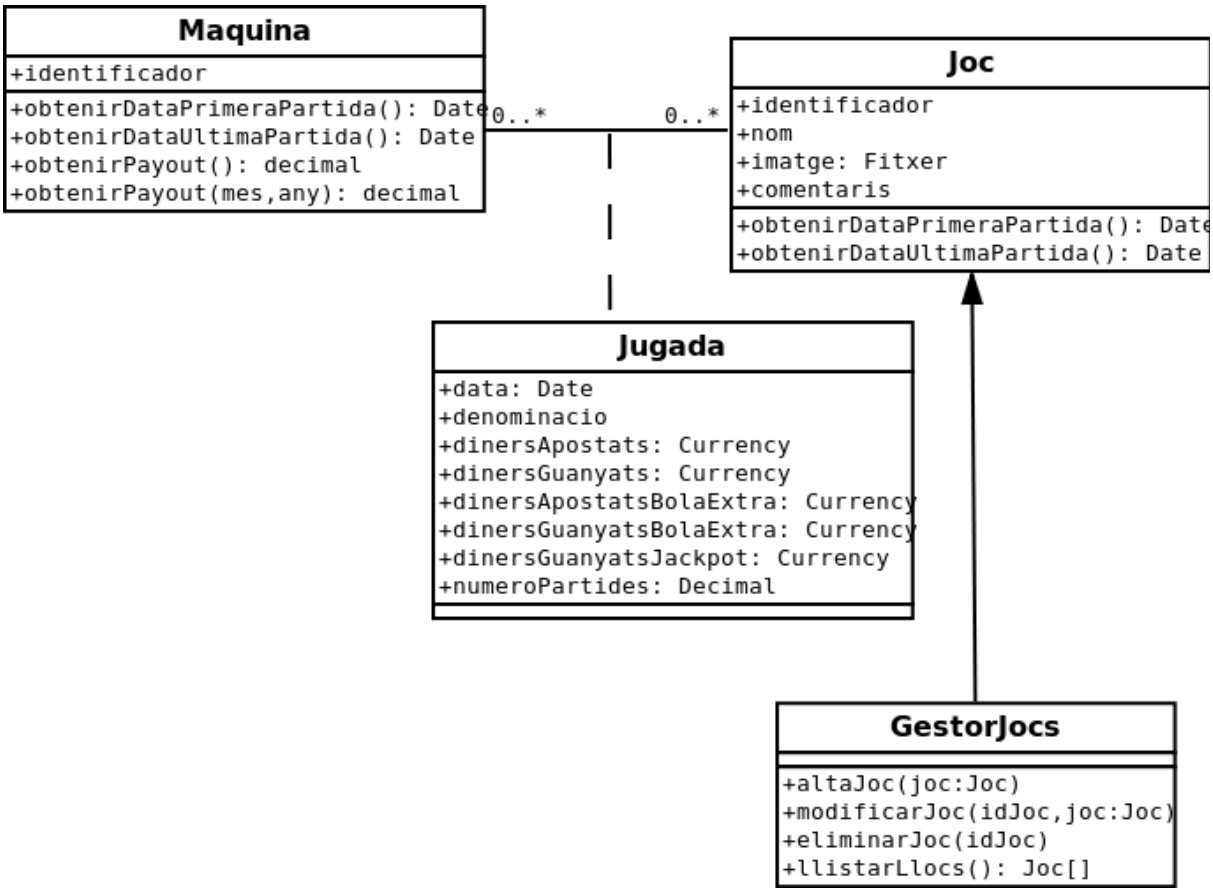


Relatius a jocs

Els casos d'ús següents:

- Alta de joc, Examinar joc, Modificar joc, Eliminar Joc, Llistar jocs

necessitaran de la intervenció dels subsistemes **Gestió de jocs, Magatzem d'imatges** i **Dades de continguts**.



La comunicació entre els subsistemes Gestió de jocs i Dades de continguts es realitzarà mitjançant sentències SQL.

Els subsistema Magatzem respondrà a dues comandes:

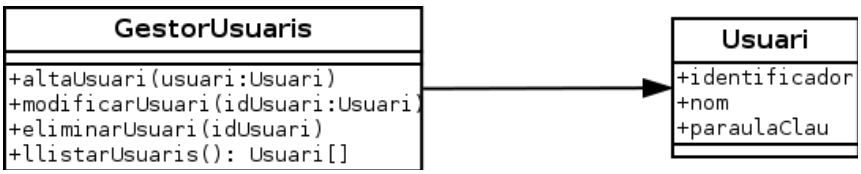
- *guardar_imatge*: rebrà com a paràmetres l'identificador de la imatge i la pròpia imatge. El retorn indicarà si l'operació s'ha pogut realitzar.
- *obtenir_imatge*: rebrà com a paràmetres l'identificador de la imatge. El retorn serà la pròpia imatge, o NULL si aquesta no existeix.

Relatius a usuaris

Els casos d'ús:

- Alta d'usuari, Modificació d'usuari, Eliminar usuari, Llistar usuaris

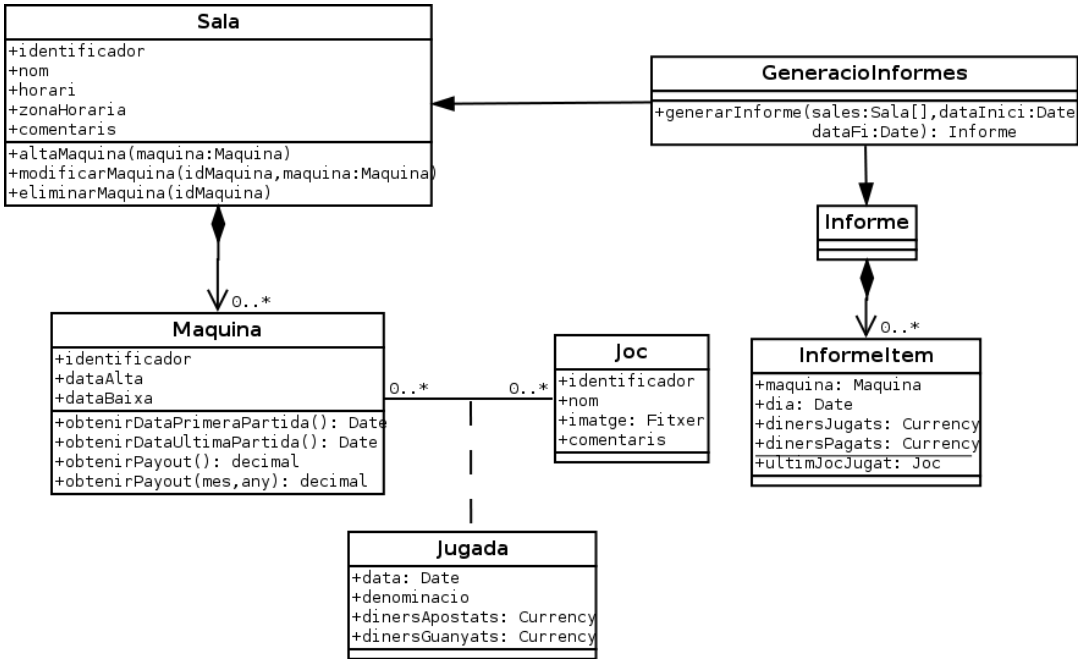
necessitaran de la intervenció del subsistemes **Gestió de sales** i **Dades de continguts**.



La comunicació entre els subsistemes es realitzarà mitjançant sentències SQL.

Generar informe

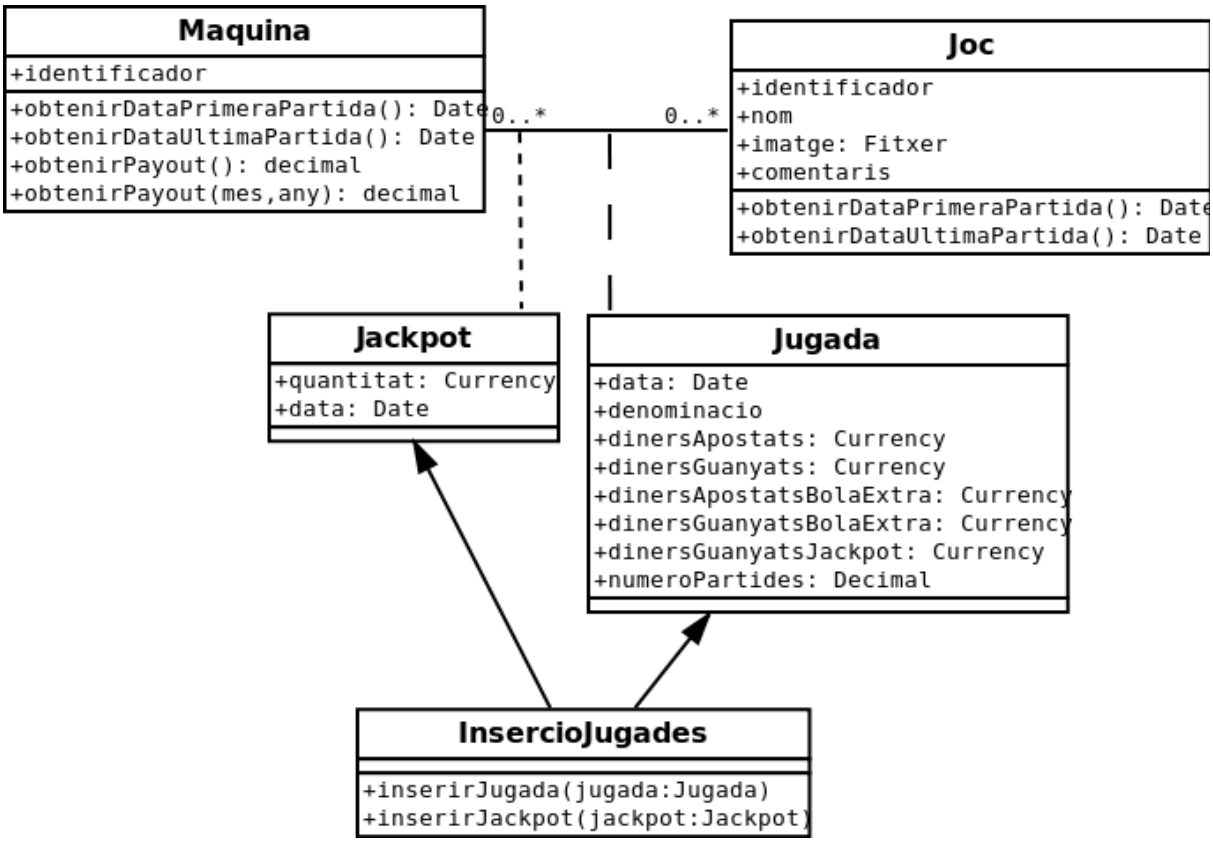
Aquest cas d'ús necessitarà de la intervenció del subsistemes **Gestió de sales** i **Dades de continguts**.



La comunicació entre els subsistemes es realitzarà mitjançant sentències SQL.

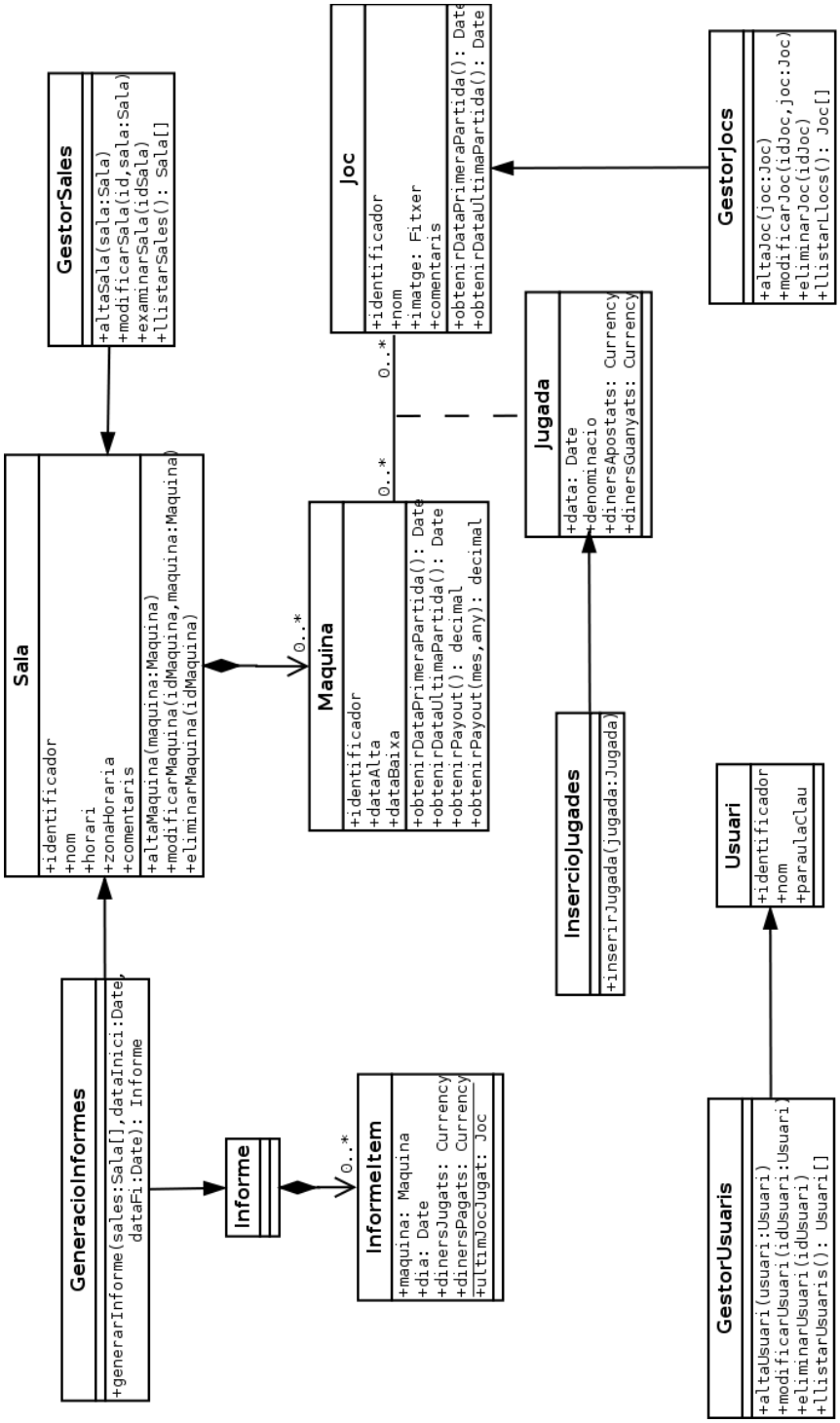
Afegir jugades a una màquina

Aquest cas d'ús necessitarà de la intervenció del subsistemes **Inserció de jugades** i **Dades de continguts**.



La comunicació entre els subsistemes es realitzarà mitjançant sentències SQL.

6.3.2. Diagrama UML de classes



6.3.3. Programari a utilitzar i llicències

Un cop analitzades els requisits i les necessitats del sistema, escolleixo els següents components de programari per al desenvolupament de l'aplicació.

Component	Paquet	Versió prevista	Llicència
Sistema operatiu	GNU/Linux	2.6.33	GPL
Servidor web	Apache	2.1.5	Apache
Bases de dades	PostgreSQL	8.4.3	BSD
Framework web	Ruby on Rails	2.2.3	MIT
Mòdul RoR per a la persistència d'imatges al sistema de fitxers	Paperclip	2.3.1.1	MIT
Mòdul RoR per Apache	Passenger	2.2.12	MIT
Mòdul RoR per PostgreSQL	Ruby-pg	0.9.9	GPL

Tots aquests components estan distribuïts sota alguna llicència lliure.

6.3.4. Descripció de l'entorn de desenvolupament

Eines

El desenvolupament es realitzarà mitjançant les següents eines:

- **Gedit**: editor de text per defecte de l'entorn d'escriptori GNOME, distribuït sota llicència GPL. S'aprofitarà la seva capacitat d'afegir extensions per tal de facilitar el desenvolupament del projecte[4]. Faré servir les següents extensions oficials:
  - **Session Saver**
  - **Snippets**
  - **File Browser Pane**
  - **Code Comment**

  A més, utilitzaré les següents extensions de terceres parts:
  - **Class Browser** (http://code.google.com/p/geditclassbrowser/): distribuït sota llicència GPL v2.
  - **Word Completion** (http://elias.hiex.at/gedit-plugins/): distribuït sota llicència GPL v2.
  - **Snapopen** (http://github.com/MadsBuus/gedit-snapopen-plugin):

  Finalment, per aconseguir la coloració del codi (*syntax highlight*) faré servir les definicions xml obtingudes de http://robzon.kapati.net/rails/rhtml.lang i http://robzon.kapati.net/rails/rails.xml.
- **Rake** (http://rake.rubyforge.org/): eina de construcció de programari equivalent a *make* però escrita en Ruby i que utilitza la sintaxi d'aquests per definir els *rakefiles* (equivalents als *makefiles*). És una eina de programari lliure llicenciada sota la llicència MIT i integrada a la biblioteca estàndar de Ruby des de la versión 1.9.

  També es pot fer servir per la execució sistèmatica de les diferents proves automàtiques.
- **Apache Subversion** (http://subversion.apache.org/): com a sistema de control de versions, de tipus centralitzat i programari lliure, llicenciat sota la llicència Apache.
- **Bugzilla** (http://www.bugzilla.org/): com a programari de gestió d'errors (*bug tracking tool*).

  És programari lliure llicenciat sota la llicència Mozilla Public License.

4 La selecció i instal·lació d'extensions es farà seguint les instruccions de la pàgina web Pimp my Gedit (Was: Textmate for Linux)

- **Mongrel** (http://github.com/fauna/mongrel): servidor web per aplicacions web Ruby, programari lliure amb llicència GPL.

  S'executa amb un únic *thread* i per això no és directament usable en producció excepte en situacions de càrrega molt lleugera, tot i que existeixen solucions per fer-lo servir juntament amb Apache. Tot i això, és ideal per al desenvolupament ja que es troba integrat amb l'entorn de Ruby on Rails.

En quant a la base de dades, faré servir PostgreSQL de la mateixa forma que en producció.

Totes aquestes eines es troben disponibles per sistemes GNU/Linux (de fet tant *rake* com *mongrel* formen part de la distribució de Ruby on Rails). De la resta, es pot trobar una versió d'aquests en els repositoris estàndards de la major part de les distribucions GNU/Linux, incloent-hi Debian.

Estil de codificació

Les convencions per defecte usades en Ruby on Rails són força simples i es redueixen a les següents[5]:

- Dos espais i no tabuladors.
- Per les operacions lògiques és millor fer servir `and` i `or` que `&&` i `||`.
- Pels mètodes millor fer servir

  ```
  MyClass.my_method(my_arg)
  ```

  i no

  ```
  my_method( my_arg )
  ```

  ni tampoc

  ```
  my_method my_arg
  ```

- Segueix les convencions que es vegis que ja es segueixen en el codi font.

A aquestes convencions afegiré les següents[6]:

- Faré servir preferentment ASCII, o UTF-8 només quan sigui necessari.
- Posaré espais al voltant dels operadors, després d'una coma i entre `{` i `}`.
- Sense espais després de `(` i `[` no abans de `)` i `]`.
- Abans del retorn d'un mètode afegiré una línia buida (excepte si el mètode només conté aquest línia).
- Afegiré línies buides en mètodes molt llargs per tal de separar blocs lògics.
- Quan sigui possible les línies hauran de ser de menys de 80 caràcters. Evitaré l'ús de `\` per marcar la continuació de línies.
- No faré servir els parèntesis en crides a mètodes si no són necessaris, però els mantindré quan cridi a funcions.

  ```
  x = Math.sin(y)
  array.delete e
  ```

- Faré servir `snake_case` pel nom dels mètodes.
- Fare servir `CamelCase` pel nom de les classes i mòduls (mantenint els acrònims com HTTP).
- Faré servir `SCREAMING_SNAKE_CASE` per les constants.

5 Extretes de http://guides.rails.info/contributing_to_rails.html#follow-the-coding-conventions, on es descriuen les convencions que s'han de seguir en el codi per tal de contribuir en el projecte.

6 Basades en les trobades a http://github.com/chneukirchen/styleguide/blob/5cb39d0935df6678ee0ab9036f5dd2056217e9a6/RUBY-STYLE.

7. Desenvolupament

Des de fa un temps, a la nostra empresa la gestió de projectes es porta amb metodologies àgils. Les metodologies àgils es defineixen per les següents característiques:

- Model de desenvolupament evolutiu en espiral, afegint noves funcionalitats a cada cicle de desenvolupament.
- Cicles de *release* de durada molt curta.
- Molta interacció amb el client final per tal de garantir ja durant el desenvolupament que el treball realitzat compleix els requisits.
- Provar abans de programar, mitjançant les proves unitàries, que permeten generar codi de millor qualitat. Totes les proves es realitzen durant tot el desenvolupament, evitant així efectes secundaris dels canvis introduïts
- Agilitat en el disseny i especificació.
- Es basa en històries d'usuari. A l'inici cada cicle es determinen les històries d'usuari que es realitzaran i al final del cicle es validen. D'aquesta manera, el desenvolupament s'enfoca a que les noves funcionalitats aportin valor al client.

El principal motiu per adaptar aquestes metodologies va ser que ofereixen flexibilitat i adaptabilitat enfront els canvis, ja que els requisits dels nostres projectes varien radicalment durant el seu desenvolupament.

De entre les metodologies àgils que existeixen, ens vam decidir per aplicar Scrum.

7.1. Scrum

La metodologia Scrum[11] està definida per una sèrie de pràctiques, rols i documents. La unitat bàsica de treball són els *Sprints*, que corresponen amb una iteració. Al començar un *Sprint* es defineixen les funcionalitats a afegir i al final es validen amb el client.

Es defineixen tres rols:

- ***Product Owner***, que representa als clients i és el responsable de definir les funcionalitats del projecte.
- ***Scrum Team***, és format pels desenvolupadors i són responsables de la codificació i organització del projecte.
- ***Scrum Master***, és responsable de garantir que els processos de l'Scrum es segueixin i de facilitar els recursos necessaris per a complir els objectius. A més, forma part de l'*Scrum Team*.

Hi ha dos documents de treball:

- ***Product Backlog***, defineix totes les funcionalitats que ha de complir l'aplicació. Aquestes funcions es corresponen amb les històries d'usuari de les metodologies àgils. Són definides pel client.
- ***Sprint Backlog***, defineix les funcionalitats que s'han d'afegir a l'aplicació durant l'*Sprint* actual.

El cicle de treball és el següent:

1. En iniciar-se el projecte, el *Product Owner* defineix el *Product Backlog* juntament amb el client i les ordena segons la prioritat de realització. Durant la vida del projecte aquestes funcionalitats es podran afegir, eliminar o modificar.
2. Un cop definit el *Product Backlog*, el primer dia de treball del projecte es fa una reunió entre el *Product Owner* i l'*Scrum Team*. El nom d'aquesta reunió és *Sprint Planning Meeting*.
    - Es defineix un periode de temps (entre una setmana i un mes) i s'escolleix la primera funcionalitat del *Product Backlog*.
    - Es pregunta a l'*Scrum Team* si creu que podran finalitzar la tasca en el plaç de temps escollit. Si cal, es pot descompondre en tasques per poder estimar millor el temps necessari. Si pensen que no podran acabar-ho, se separa en tasques més senzilles fins que poden acceptar acabar alguna en temps.

- A continuació s'escolleix la següent tasca i es pregunta a l'*Scrum Team* si poden fer ambdues en el temps definit.

Aquest procés es repeteix amb les següents tasques fins que no en poden afegir més. El *Product Owner* pot decidir afegir tasques, intercanviar-les, modificar-les o partir-les, però l'*Scrum Team* té la darrera paraula per acceptar-ho només si pensa que ho podrà finalitzar en el temps.

Una vegada arriben a un acord, les funcionalitats escollides passen a formar l'*Sprint Backlog*. En el temps indicat s'entregarà al *Product Owner* una versió del programa amb totes las funcionalitats seleccionades funcionant.

3. Cada dia a primera hora es fa una reunió anomenada *Daily Scrum Meeting*, a la qual hi participa l'*Scrum Team*. Es fan tres preguntes a cada membre:
   - Què vas fer ahir?
   - Què faràs avui?
   - Quina ajuda necessites?

   En aquesta reunió, l'*Scrum Master* té dues funcions. La primera és evitar que duri més de 15-30 minuts i la segona, aconseguir les ajudes necessàries.

   En aquesta reunió es torna a estimar el temps de les tasques en curs, per tal de determinar si la planificació s'està complir.
4. En finalitzar l'*Sprint* es fa una reunió anomenada *Sprint Review*, a la qual assistiran totes les persones implicades en el projecte. L'*Scrum Master* ensenya la versió i els assistents poden donar opinions o proposar possibles millores que s'afegiran al *Product Backlog*.
5. A continuació es fa una nova reunió (anomenada *Sprint Restropective*) en la que l'*Scrum Master*, l'*Scrum Team* y el *Product Owner* analitzen quines millores es poden afegir en la forma de treballar i si hi ha algun problema per solucionar pensant en el següent *Sprint*.
6. Per finalitzar, s'inicia un nou *Sprint*, realitzant un *Sprint Planning Meeting*.

Aquest procés continuarà fins que totes les tasques estiguin realitzades i tinguin l'aprovació del client.

7.2. Aplicació de la metodologia al projecte

En general, hem intentat seguir la metodologia Scrum tant com ens ho han permès les particularitats del projecte. Respecte a l'explicació general, hem realitzat les següents modificacions:

- En el nostre cas, l'*Scrum Team* ha estat format per una sola persona, que per tant també és l'*Scrum Master*. Per això, no s'han realitzat les reunions diàries.
- A més de l'*Scrum Team*, hi ha hagut dues persones més implicades al projecte. D'una banda, tenim el *Product Owner*, que en aquest cas és un altre treballador de l'empresa que serà a més l'encarregat de fer servir l'aplicació. De l'altra, una altre treballador ha estat l'encarregat de fer les proves de l'aplicació.
- La resta de reunions sí s'han realitzat i han participat les tres persones implicades en el projecte.

Respecte a la durada dels S*prints*, vam aprofitar que les tasques eren força curtes i es van acordar d'una setmana. A cada S*print* es van tornar a passar totes les proves unitàries i d'acceptació.

7.2.1. Product Backlog

De la primera reunió de definició del projecte va quedar el *Product Backlog* definit per les següents històries d'usuari:

1. Es poden afegir, consultar, modificar i eliminar jocs.
2. Es poden afegir, consultar, modificar i eliminar sales.
3. S'afegeixen jugades mitjançant un servei web.
4. Les dades antigues han d'estar disponibles en el sistema des de l'inici.

5. S'han de generar informes de dades acumulades. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra, número de partides i número de màquines.
6. Les dades dels informes s'han de poder exportar en format *csv*.
7. Les pàgines han de tenir un disseny comú.
8. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.

A aquestes primeres històries d'usuari es van afegir d'altres durant el desenvolupament, a mesura que els requeriments es modificaven.

1. S'afegeixen *jackpots* mitjançant un servei web.
2. Es poden consultar els *jackpots* de cada sala.
3. S'han de generar informes dels *jackpots*. Es poden filtrar per sala i data. Serà possible consultar els darrers 10, 20, 50 i 100. De cada *jackpot* es mostrarà el nom de la sala, el codi de la màquina, el nom del joc, la data i la quantitat.
4. S'han de generar informes històrics, agrupats per setmanes. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són data, crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra i número de partides.
5. En consultar una sala, s'han de veure les màquines existents i de cadascuna d'aquestes les següents dades: identificador, primera jugada i joc, última jugada i joc.
6. En consultar una sala, s'han de veure els últims 4 *jackpots* en qualsevol màquina.
7. En consultar un joc, s'han de veure les següents dades: primera i última jugada, número total de partides i percentatge de pagament total, sense bola extra i només bola extra.

7.2.2. Sprints

Per mostrar l'evolució del procés de desenvolupament apareix a continuació el resultat de cada reunió inicial d'*Sprint*. Es mostra l'*Spring Backlog* i les tasques pendents que queden al *Product Backlog*.

Sprint 1	**1/11/10 fins al 7/11/10**
Sprint Backlog	1. Es poden afegir, consultar, modificar i eliminar jocs. 2. Les pàgines han de tenir un disseny comú.
Product Backlog	3. Es poden afegir, consultar, modificar i eliminar sales. 4. Les dades antigues han d'estar disponibles en el sistema des de l'inici. 5. S'afegeixen jugades mitjançant un servei web. 6. S'han de generar informes de dades acumulades. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra, número de partides i número de màquines. 7. Les dades dels informes s'han de poder exportar en format csv. 8. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.

Sprint 2	**8/11/10 fins al 14/11/10**
Sprint Backlog	1. Es poden afegir, consultar, modificar i eliminar sales.
Product Backlog	2. Les dades antigues han d'estar disponibles en el sistema des de l'inici. 3. S'afegeixen jugades mitjançant un servei web. 4. S'han de generar informes de dades acumulades. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són crèdits jugats, credits pagats, crèdits de recaudació, percentatge de pagament, crèdits jugats i guanyats en bola extra, número de partides i número de màquines. 5. Les dades dels informes s'han de poder exportar en format csv. 6. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.

Sprint 3	**15/11/10 fins al 21/11/10**
Sprint Backlog	1. Les dades antigues han d'estar disponibles en el sistema des de l'inici.
Product Backlog	2. S'afegeixen jugades mitjançant un servei web. 3. En consultar una sala, s'han de veure les màquines existents i de cadascuna d'aquestes les següents dades: identificador, primera jugada i joc, última jugada i joc. 4. En consultar un joc, s'han de veure les següents dades: primera i última jugada, número total de partides i percentatge de pagament total, sense bola extra i només bola extra. 5. S'han de generar informes de dades acumulades. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra, número de partides i número de màquines. 6. Les dades dels informes s'han de poder exportar en format *csv*. 7. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.
Comentaris	Al final d'aquest *Sprint* es va decidir afegir noves funcionalitats, que corresponen als punts 3 i 4 del *Product Backlog*.

Sprint 4	**22/11/10 fins al 28/11/10**
Sprint Backlog	1. S'afegeixen jugades mitjançant un servei web. 2. En consultar una sala, s'han de veure les màquines existents i de cadascuna d'aquestes les següents dades: identificador, primera jugada i joc, última jugada i joc. 3. En consultar un joc, s'han de veure les següents dades: primera i última jugada, número total de partides i percentatge de pagament total, sense bola extra i només bola extra.
Product Backlog	4. S'han de generar informes de dades acumulades. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra, número de partides i número de màquines. 5. Les dades dels informes s'han de poder exportar en format *csv*. 6. El control d'usuaris s'ha de poder gestionar des de l'aplicació.

Sprint 5	**29/11/10 fins al 5/12/10**
Sprint Backlog	1. S'han de generar informes de dades acumulades. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra, número de partides i número de màquines. 2. Les dades dels informes s'han de poder exportar en format *csv*.
Product Backlog	3. S'afegeixen *jackpots* mitjançant un servei web. 4. En consultar una sala, s'han de veure els últims 4 *jackpots* en qualsevol màquina. 5. Es poden consultar els *jackpots* de cada sala. 6. S'han de generar informes dels *jackpots*. Es poden filtrar per sala i data. Serà possible consultar els darrers 10, 20, 50 i 100. De cada *jackpot* es mostrarà el nom de la sala, el codi de la màquina, el nom del joc, la data i la quantitat. 7. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.
Comentaris	Al final d'aquest *Sprint* es va decidir afegir noves funcionalitats, que corresponen als punts 3, 4, 5 i 6 del *Product Backlog*.

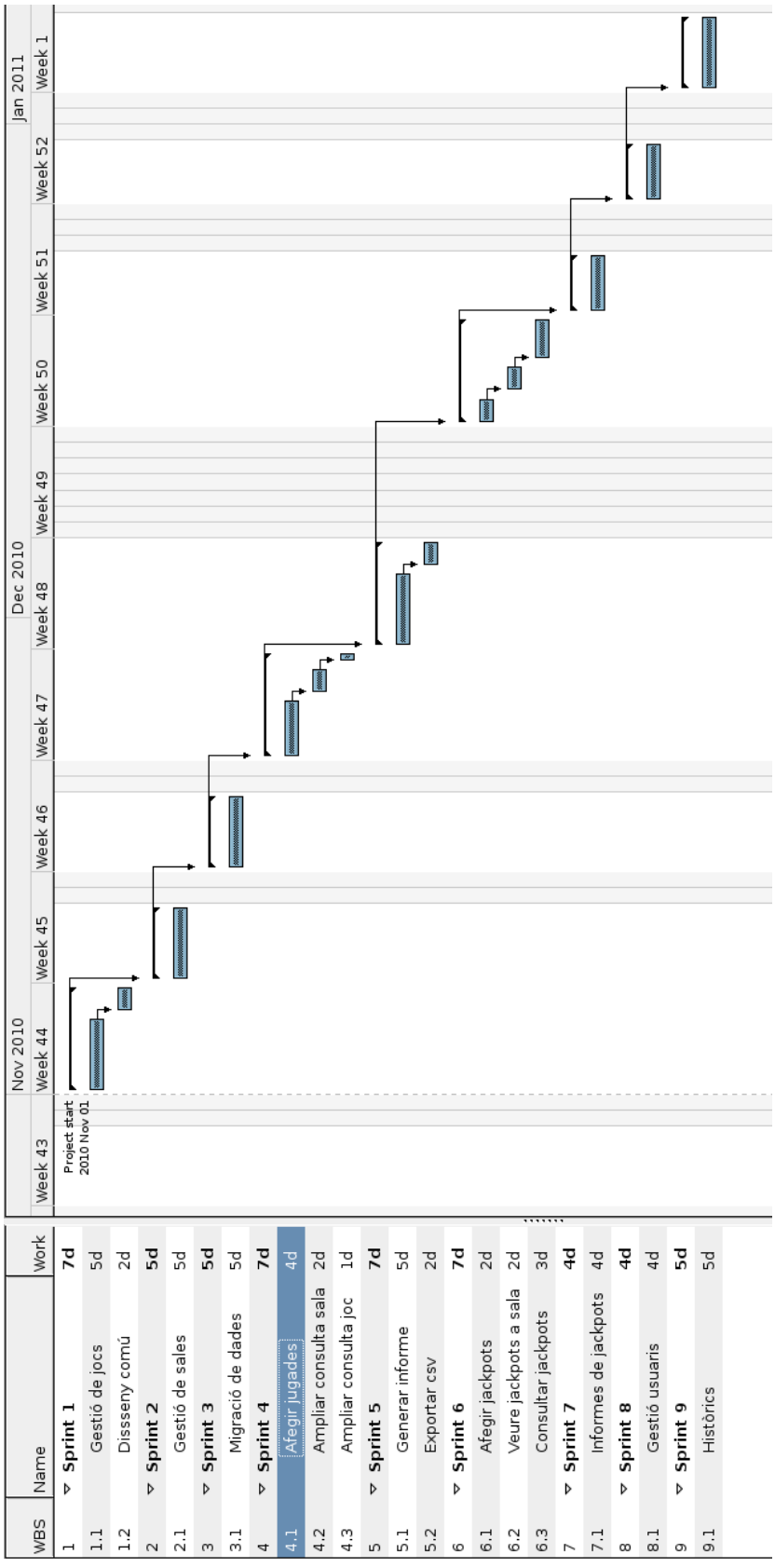
Sprint 6	**13/12/10 fins al 19/12/10**
Sprint Backlog	1. S'afegeixen *jackpots* mitjançant un servei web. 2. En consultar una sala, s'han de veure els últims 4 *jackpots* en qualsevol màquina. 3. Es poden consultar els *jackpots* de cada sala.
Product Backlog	4. S'han de generar informes dels *jackpots*. Es poden filtrar per sala i data. Serà possible consultar els darrers 10, 20, 50 i 100. De cada *jackpot* es mostrarà el nom de la sala, el codi de la màquina, el nom del joc, la data i la quantitat. 5. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.
Comentaris	Entre aquest Sprint i l'anterior no es va continuar el desenvolupament, degut al pont del 6 i 8 de desembre.

Sprint 7	**20/12/10 fins al 26/12/10**
Sprint Backlog	1. S'han de generar informes dels *jackpots*. Es poden filtrar per sala i data. Serà possible consultar els darrers 10, 20, 50 i 100. De cada *jackpot* es mostrarà el nom de la sala, el codi de la màquina, el nom del joc, la data i la quantitat.
Product Backlog	2. S'han de generar informes històrics, agrupats per setmanes. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són data, crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra i número de partides. 3. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.
Comentaris	Al final d'aquest *Sprint* es va decidir afegir una nova funcionalitat, que corresponen al punt 2 del *Product Backlog*.

Sprint 8	**27/12/10 fins al 30/12/10**
Sprint Backlog	1. El control d'usuaris s'ha de poder gestionar des de la mateixa aplicació.
Product Backlog	2. S'han de generar informes històrics, agrupats per setmanes. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són data, crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra i número de partides.
Comentaris	En la reunió inicial d'aquest *Sprint* es va decidir realitzar primer la tasca 3 de l'anterior *Product Backlog*, ja que era una setmana curta deguda al Cap d'Any.

Sprint 9	**3/1/11 fins al 9/12/11**
Sprint Backlog	1. S'han de generar informes històrics, agrupats per setmanes. Es poden agrupar per joc i sala. Es poden filtrar per joc, sala i dates. Les dades a mostrar són data, crèdits jugats, crèdits pagats, crèdits de recaptació, percentatge de pagament, crèdits jugats i guanyats en bola extra i número de partides.
Product Backlog	-
Comentaris	Aquest va ser l'últim Sprint del desenvolupament.

7.2.3. Diagrama de Gantt



7.3. Organització del codi font

Ruby on Rails ofereix un entorn de desenvolupament que es basa en dos conceptes: no repetir-se (en anglès, *DRY – Don't Repeat Yourself*) i convenció sobre configuració.

El concepte de convenció sobre configuració significa que la major part dels aspectes de l'aplicació tenen uns valors per defecte. Aquests valors es poden modificar, però no és recomanat. D'aquesta forma, l'estructura d'un projecte Rails generalment tindrà sempre la mateixa estructura.

Se segueix el patró model-vista-controlador, com queda reflectit en l'estructura de fitxers.

El codi font de l'aplicació està composat pels següents directoris i fitxers:

./app/controllers: add_gamble_info_controller.rb add_jackpot_controller.rb admin_controller.rb application.rb games_controller.rb halls_controller.rb historicals_controller.rb jackpots_controller.rb reports_controller.rb users_controller.rb	Conté els controladors.
./app/helpers: add_gamble_info_helper.rb add_jackpot_helper.rb admin_helper.rb application_helper.rb gamble_info_helper.rb games_helper.rb halls_helper.rb jackpots_helper.rb reports_helper.rb users_helper.rb	Conté classes que ofereixen mètodes comuns a tots els components del patró model-vista-controlador.
./app/models: gamble_info.rb game.rb hall.rb historical_item.rb historical.rb jackpot.rb report_item.rb report.rb user.rb	Conté els models.
./app/views/admin: login.html.erb	Conté les vistes d'administració.
./app/views/games: edit.html.erb index.html.erb new.html.erb show.html.erb	Conté les vistes relatives als jocs.
./app/views/halls: edit.html.erb index.html.erb new.html.erb show.html.erb	Conté les vistes relatives a les sales.
./app/views/historicals: _historical.html.erb new.html.erb update.js.rjs	Conté les vistes relatives als històrics.

./app/views/jackpots: index.html.erb _jackpots.html.erb list_jackpots.js.rjs	Conté les vistes relatives als *jackpots*.
./app/views/layouts: application.html.erb	Conté el layout general de l'aplicació.
./app/views/reports: new.html.erb _report.html.erb update.js.rjs	Conté les vistes relatives als informes.
./app/views/users: edit.html.erb index.html.erb new.html.erb show.html.erb	Conté les vistes relatives als usuaris.
./db/migrate 20100905204815_create_games.rb 20100905204840_add_image_to_game.rb 20100906180822_add_comments_to_game.rb 20100906190747_create_halls.rb 20101114181219_create_gamble_infos.rb 20101210211159_create_users.rb 20101213201629_add_users.rb 20101220215828_create_jackpots.rb	Conté els scripts de generació de la base de dades.
./public/images: logo.png rails.png **./public/images/16:** add.png button_cancel.png button_ok.png edit.png remove.png view.png **./public/images/32:** add.png back.png button_cancel.png button_ok.png edit.png remove.png	Conté les imatges emprades en l'aplicació.
./public/stylesheets: application.css scaffold.css	Conté les fulles d'estil *css* de l'aplicació.

7.4. Accés als serveis web

Finalment han quedat definits dos serveis web. El servei web per inserir jugades queda definit de la següent forma:

URL	https://dev.ludicus.com:8029/add_gamble_info	
Mètode	POST	
Paràmetres	gamble_info[hall_code]	Codi de la sala.
	gamble_info[machine_code]	Codi de la màquina.
	gamble_info[game_code]	Codi del joc.
	gamble_info[date]	Data de les dades.
	gamble_info[cr_in]	Crèdits jugats totals.
	gamble_info[cr_out]	Crèdits guanyats totals.
	gamble_info[cr_in_be]	Crèdits jugats en bola extra.
	gamble_info[cr_out_be]	Crèdits guanyats en bola extra.
	gamble_info[cr_out_jackpot]	Crèdits guanyats en per *jackpot*.
	gamble_info[denomination]	Denominació a la qual s'han jugat les partides.
	gamble_info[num_gambles]	Número de partides.

Per provar-ho, es pot fer servir la comanda curl, com es mostra en el següent exemple:

add_gamble_info.sh

```
#!/bin/bash

TRG_URL=https://dev.ludicus.com:8029/add_gamble_info

curl -u machine:entrar -k -X POST ${TRG_URL} -d
"gamble_info[hall_code]=lombarda&gamble_info[machine_code]=1&gamble_info[game
_code]=bichos&gamble_info[date]=1222002000000&gamble_info[cr_in]=10000&gamble
_info[cr_out]=0&gamble_info[cr_in_be]=5000&gamble_info[cr_out_be]=0&gamble_in
fo[cr_out_jackpot]=0&gamble_info[denomination]=100&gamble_info[num_gambles]=2
"
```

El resultat de la crida és una descripció de la jugada introduida, en format xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<gamble-info>
  <cr-in type="decimal">10000.0</cr-in>
  <cr-in-be type="decimal">5000.0</cr-in-be>
  <cr-out type="decimal">0.0</cr-out>
  <cr-out-be type="decimal">0.0</cr-out-be>
  <cr-out-jackpot type="decimal">0.0</cr-out-jackpot>
  <created-at type="datetime">2011-01-20T19:54:01Z</created-at>
  <date type="decimal">1222002000000.0</date>
  <denomination type="decimal">100.0</denomination>
  <game-code>bichos</game-code>
  <hall-code>lombarda</hall-code>
  <id type="integer">189006</id>
  <machine-code type="decimal">1.0</machine-code>
  <num-gambles type="decimal">2.0</num-gambles>
  <updated-at type="datetime">2011-01-20T19:54:01Z</updated-at>
</gamble-info>
```

Una crida errònia retornarà un xml amb informació sobre què ha fallat. En el cas següent, no s'ha enviat el número de partides:

```
<?xml version="1.0" encoding="UTF-8"?>
<errors>
  <error>Num gambles can't be blank</error>
  <error>Num gambles is not a number</error>
</errors>
```

El servei web per inserir *jackpots* queda definit de la següent forma:

URL	https://dev.ludicus.com:8029/add_jackpot	
Mètode	POST	
Paràmetres	jackpot[hall_code]	Codi de la sala.
	jackpot[machine_code]	Codi de la màquina.
	jackpot[game_code]	Codi del joc.
	jackpot[date]	Data de les dades.
	jackpot[amount]	Crèdits guanyats.

Per provar-ho, es pot fer servir la comanda curl, com es mostra en el següent exemple:

add_jackpot.sh

```
#!/bin/bash

TRG_URL=https://dev.ludicus.com:8029/add_jackpot

curl -u machine:entrar -k -X POST ${TRG_URL} -d
"jackpot[hall_code]=ganaygana&jackpot[machine_code]=1&jackpot[game_code]=splo
sh&jackpot[date]=111111111111111&jackpot[amount]=3000"
```

El resultat de la crida és una descripció del jackpot introduït, en format xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<jackpot>
  <amount type="decimal">3000.0</amount>
  <created-at type="datetime">2011-01-20T19:49:40Z</created-at>
  <date type="decimal">111111111111111.0</date>
  <game-code>splosh</game-code>
  <hall-code>ganaygana</hall-code>
  <id type="integer">755</id>
  <machine-code type="decimal">1.0</machine-code>
  <updated-at type="datetime">2011-01-20T19:49:40Z</updated-at>
</jackpot>
```

Una crida errònia retornarà un xml amb informació sobre què ha fallat. En el cas següent, no s'ha enviat la quantitat guanyada:

```
<?xml version="1.0" encoding="UTF-8"?>
<errors>
  <error>Amount can't be blank</error>
  <error>Amount is not a number</error>
</errors>
```

7.5. Proves

S'han definit tres tipus de proves:

- Proves unitàries i funcionals, que s'han realitzat juntament amb la codificació usant les eines que ofereix Ruby on Rails. Al finalitzar cada *Sprint* es tornaven a realitzar totes aquestes proves per garantir que els canvis introduïts no provocaven efectes col·laterals, per part de l'*Scrum Master*.
- Proves d'acceptació, que es realitzaven manualment al final de cada *Sprint* per la persona encarregada. Les proves es definien al final de cada *Sprint*, afegint les necessàries per comprovar les funcionalitats afegides.

7.5.1. Proves unitàries

Es poden trobar en el codi font de l'aplicació, a la carpeta test. Els fitxers són els següents:

`test/fixtures/gamble_infos.yml`	Dades inicials per a les proves unitàries i funcionals, referents a les jugades.
`test/fixtures/games.yml`	Com l'anterior, referents als jocs.
`test/fixtures/halls.yml`	Com l'anterior, referents a les sales.
`test/fixtures/jackpots.yml`	Com l'anterior, referents als *jackpots*.
`test/fixtures/users.yml`	Com l'anterior, referents als usuaris.
`test/functional/games_controller_test.rb`	Proves funcionals sobre els jocs.
`test/functional/halls_controller_test.rb`	Proves funcionals sobre les sales.
`test/functional/jackpots_controller_test.rb`	Proves funcionals sobre els *jackpots*.
`test/functional/users_controller_test.rb`	Proves funcionals sobre els usuaris.
`test/unit/gamble_info_test.rb`	Proves unitàries sobre les jugades.
`test/unit/game_test.rb`	Proves unitàries sobre els jocs.
`test/unit/hall_test.rb`	Proves unitàries sobre les sales.
`test/unit/jackpot_test.rb`	Proves unitàries sobre els *jackpots*.
`test/unit/report_test.rb`	Proves unitàries sobre els informes.
`test/unit/user_test.rb`	Proves unitàries sobre els usuaris.

Aquestes proves s'executen en bloc mitjançant la comanda `rake`, disponible per defecte en instal·lar Ruby on Rails.

```
rake test
```

7.5.2. Proves d'acceptació

El guió final de la prova d'acceptació ha estat el següent:

Prova d'acceptació	
Provador	**Data**

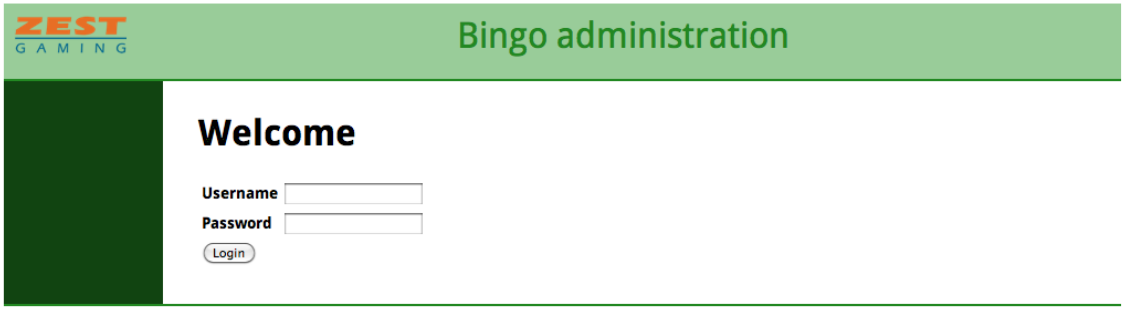
Descripció	**OK**	**Error**
Accedir a https://dev.ludicus.com:8029 ens porta a la pantalla de login.		
Accedir amb l'usuari 'david' i el password 'entrar1' no permet accedir a l'aplicació		
Accedir amb l'usuari 'david' i el password 'entrar' permet accedir a l'aplicació		
A la barra lateral, al costat de l'opció Logout apareix el nom de l'usuari (en aquest cas, [david])		

Accedeix al llistat d'usuaris. No apareix la opció d'eliminar l'usuari 'david'.		
Edita l'usuari 'david' i canvia el seu password per 'entrar1'. Fes logout i torna a fer login amb el password 'entrar'. No deixa accedit a l'aplicació.		
Fes login amb el password 'entrar1'. Es permet l'accés a l'aplicació.		
Crea l'usuari 'default' amb password 'default'. Fes logout i després login amb aquest nou usuari. Es permet l'accés a l'aplicació.		
A la barra lateral, al costat de l'opció Logout apareix el nom de l'usuari (en aquest cas, [default])		
Des del llistat d'usuaris, elimina l'usuari 'david'. Ja no apareix al llistat.		
Fes logout i accedeix amb l'usuari 'david' i el password 'entrar1'. No es permet l'accés.		
Accedeix amb l'usuari ' default' i el password ' default'. Es permet l'accés.		
Amb la opció 'New hall' crea una nova sala amb les següents dades: • code: test • name: Test Hall • Open: 10h • Close: 2h • Timezone: (GMT-06:00) Mexico City La sala es crea correctament.		
Accedeix al llistat de sales. Apareix la sala 'Test Hall', sense partides jugades ni *jackpots*.		
Amb la opció 'New hall' crea una nova sala amb les següents dades: • code: test • name: Test Hall 2 • Open: 10h • Close: 2h • Timezone: (GMT-06:00) Mexico City Es produeix l'error '`Code has already been taken`' i la sala no es crea. No apareix a la llista de sales.		
Elimina la sala 'Test Hall'. La sala deixa d'aparèixer al llistat de sales.		
Amb la opció 'New game' crea un nou joc amb les dades següents: • code: test • name: Test Game • image: Qualsevol imatge El joc es crea correctament.		
Accedeix al llistat de jocs. Apareix el joc 'Test Game' mostrant com a logo la imatge escollida.		
Elimina el joc 'Test Game'. El joc deixa d'aparèixer al llistat de jocs.		
Genera un informe mitjançant la opció *Report*. Apareixen els següents camps: *Credit in, Credit out, Win, Payout, Credit in extra, Credit out extra, Gambles, Machines*.		
Genera un nou informe agrupant per sala. Apareixen els camps anteriors més el camp *Hall*.		
Genera un nou informe agrupant per sala i joc. Apareixen els camps anteriors més el camp *Game*.		
Genera un històric mitjançant la opció *Historical*. Apareixen els següents camps: *Date, Credit in, Credit out, Win, Payout, Credit in extra, Credit out extra, Gambles*.		

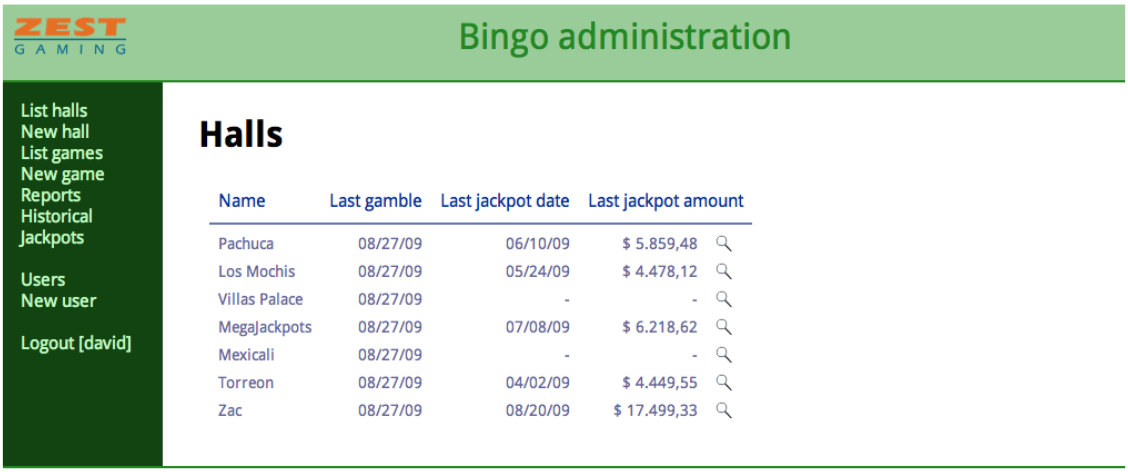
Genera un nou històric agrupant per sala. Apareixen els camps anteriors més el camp *Hall*.		
Genera un nou històric agrupant per sala i joc. Apareixen els camps anteriors més el camp *Game*.		
Genera un llistat de *jackpots* mitjançant la opció *Jackpots*. Apareixen els següents camps: *Hall, Machine, Game, Date, Amount*.		

7.6. Captures de pantalla

A continuació es mostra el resultat final del desenvolupament.

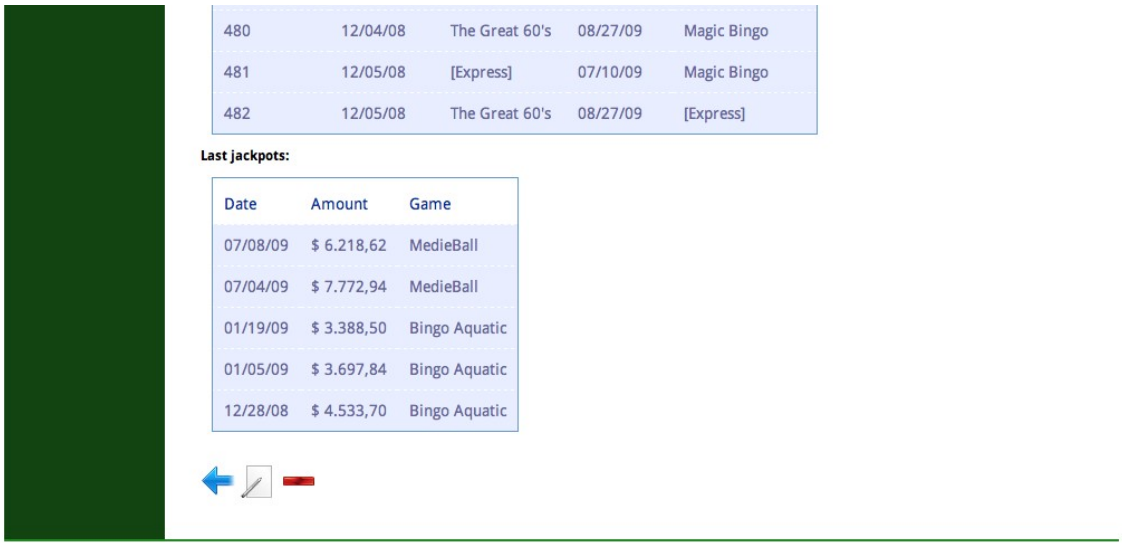


Login

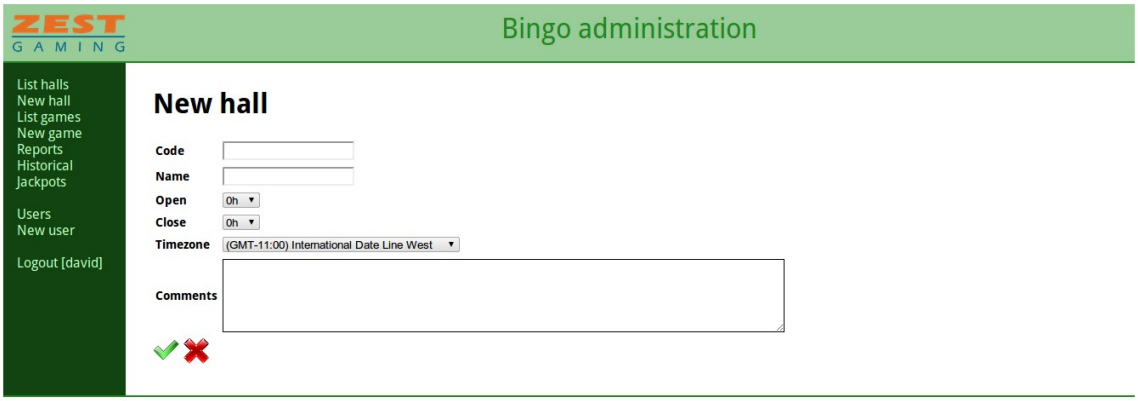


Llistat de sales

Examinar sala (1)

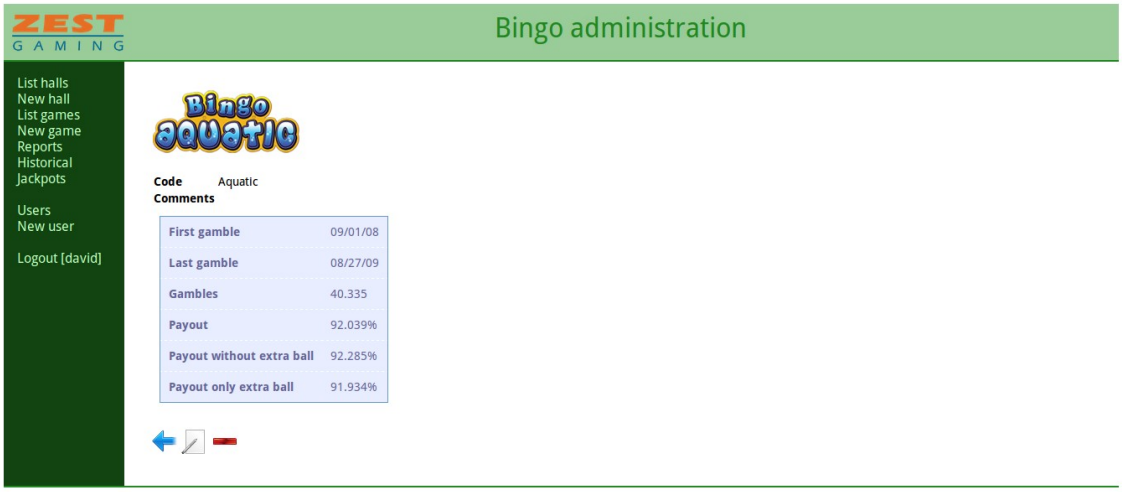


Examinar sala (2)

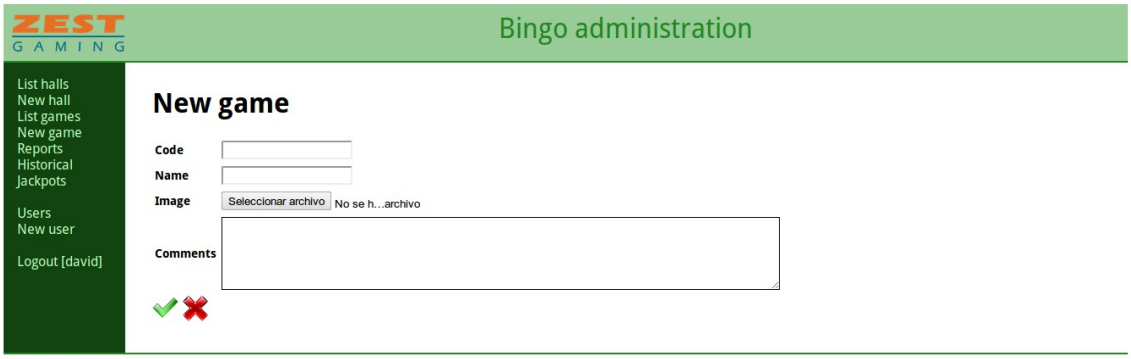


Nova sala

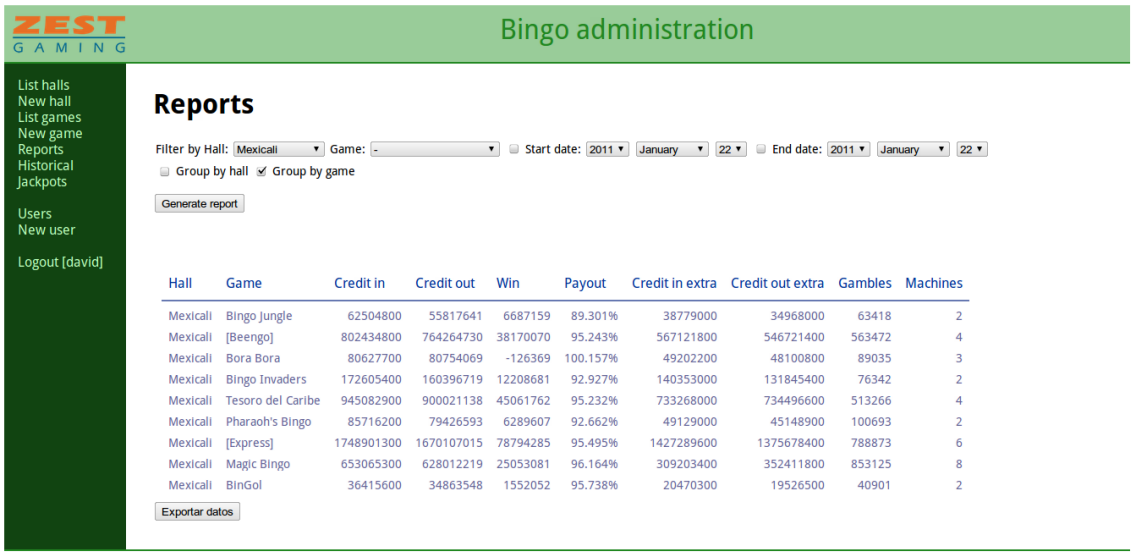
Llistat de jocs



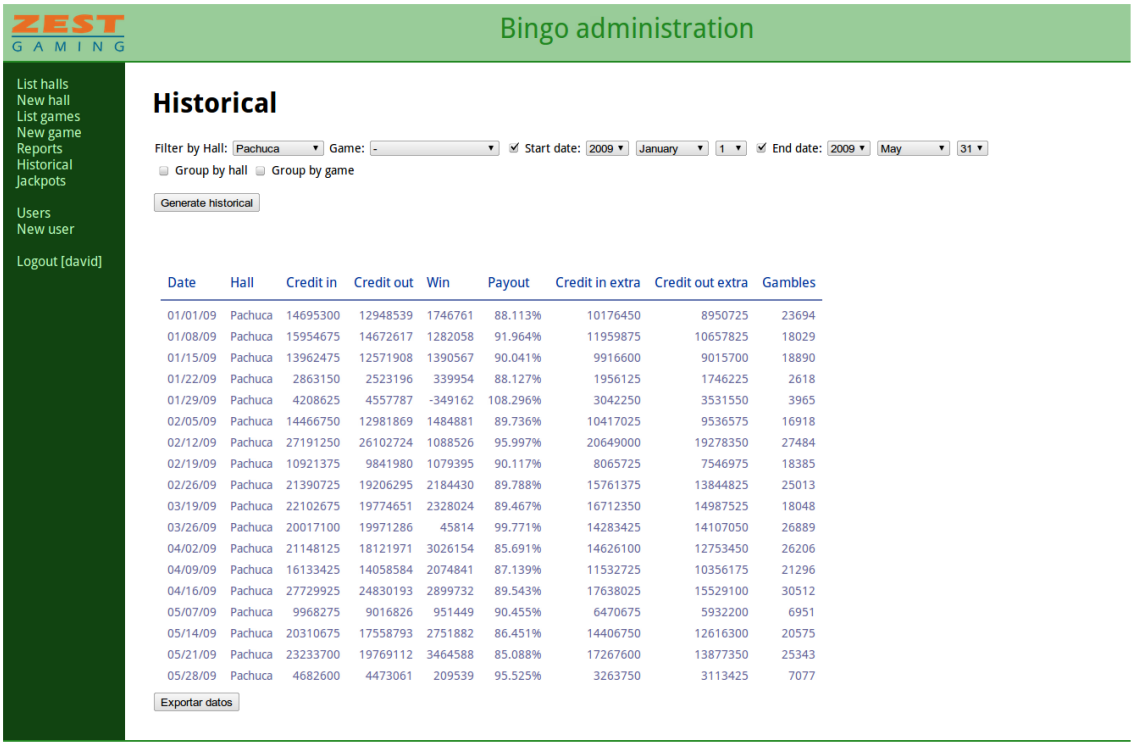
Examinar joc



Nou joc



Informe, filtrant per sala i agrupant per joc



Històric, filtrant per sala i data

Llistat de *jackpots*, filtrant per sala

8. Implantació

El procés d'implantació ha estat paral·lel al procés de desenvolupament, ja que totes les proves d'acceptació s'han fet utilitzant el servidor que es farà servir en producció.

Ben aviat durant el procés de desenvolupament (a l'*Sprint* número 3) es van introduir les dades reals que s'havien obtingut de les màquines de bingo durant els darrers 3 anys, permetent des d'un principi la detecció i resolució de problemes en passar a producció.

Això a més ens ha permès modificar els requisits de l'aplicació amb les suggerències dels propis usuaris, fent que el resultat final sigui més adient a les necessitats de l'empresa.

8.1. Instal·lació del servidor web

El servidor que es fa servir a l'aplicació ja tenia instal·lat un servidor web Apache i un servidor de bases de dades PostgreSQL. Per tant, només ha estat necessari instal·lar el propi Ruby on Rails [12] i els plug-ins necessaris per la seva connexió amb ambdós servidors [13][14].

8.1.1. Control d'accés

Totes les comunicacions amb el servidor Apache estan xifrades mitjançant SSL. Els accesos a l'aplicació web requereixen d'identificació des de la pròpia aplicació, no sent així en el cas de l'accés als serveis web.

Per als serveis web es va decidir aprofitar el mecanisme d'autenticació del propi Apache [15]. La configuració del lloc web queda així:

/etc/apache2/sites-available/default

```
...
<VirtualHost *:443>
  ServerName dev.ludicus.com
  DocumentRoot /home/bingo/public
  <Directory /home/bingo/public>
  AllowOverride all
  Options -MultiViews
  </Directory>
  <Location /add_jackpot>
  AuthType Basic
  AuthName "Restricted"
  AuthUserFile /home/bingo/passwords
  Require user machine
  </Location>

  <Location /add_gamble_info>
  AuthType Basic
  AuthName "Restricted"
  AuthUserFile /home/bingo/passwords
  Require user machine
  </Location>

  SSLEngine on
  SSLCertificateFile /etc/apache2/ssl/apache.pem
</VirtualHost>
```

8.1.2. Migració de les dades

Les dades existents consistien en les jugades fetes en totes les màquines durant els últims tres anys. Aquestes dades ja contenien totes les columnes que es requerien i només ha estat necessari modificar el nom de la taula i dels camps per que coincideixin amb les emprades per defecte per Ruby on Rails.

El cas dels jackpots és diferent. No hi havia cap taula amb dades dels *jackpots*, però aquestes han estat fàcilment extretes a partir de les dades de les partides. Hem assumit que si un registre de jugades contenia diners guanyats en el *jackpot*, aquests corresponien només a un. Donat que la freqüència dels jackpots és més gran que un dia, aquesta suposició és sempre certa.

Per transferir les dades, s'ha fet servir el següent script, que ataca contra el servei web d'inserció de *jackpots*:

transfer_jackpots.sh

```
#!/bin/bash

SRC_DB_HOST=localhost
SRC_DB_NAME=bingo
SRC_DB_USER=bingo

TRG_URL=https://localhost:443/add_jackpot

for row in `psql -A -t -h ${SRC_DB_HOST} -d ${SRC_DB_NAME} -U ${SRC_DB_USER}
-c "SELECT hall_code,machine_code,game_code,date,cr_out_jackpot FROM
gamble_infos WHERE cr_out_jackpot>0;"`; do
 HALL_CODE=`echo $row | cut -d '|' -f 1`
 MACHINE_CODE=`echo $row | cut -d '|' -f 2`
 GAME_CODE=`echo $row | cut -d '|' -f 3`
 DATE=`echo $row | cut -d '|' -f 4`
 AMOUNT=`echo $row | cut -d '|' -f 5`
 curl -u machine:entrar -k -X POST ${TRG_URL} -d "jackpot[hall_code]=$
{HALL_CODE}&jackpot[machine_code]=${MACHINE_CODE}&jackpot[game_code]=$
{GAME_CODE}&jackpot[date]=$DATE&jackpot[amount]=$AMOUNT"
done
```

8.1.3. Seguretat

Donat que el servidor ja estava en ús en el moment d'implantar el projecte, no ha calgut modificar les polítiques de seguretat i es continuaran fent servir les existents.

El mateix s'aplica a les còpies de seguretat. El protocol existent ja realitzava còpies de seguretat de les bases de dades allotjades al servidor, ara incloent les de l'aplicació.

9. Conclusions

En el moment de finalitzar el projecte s'han assolit satisfactòriament tots els objectius desitjats en el temps disponible. L'aplicació està finalitzada, els serveis web disponibles i ja s'està fent servir per l'estudi de les dades provinents de les màquines de Mèxic.

Actualment s'estan modificant els terminals de joc per tal que enviïn automàticament les dades un cop al dia emprant el servei web corresponent, així com els *jackpots* guanyats. La càrrega de treball no serà massa gran, ja que l'emmagatzemament de les dades en local es realitza des de l'inici de l'operació.

Tot i així, s'han plantejat algunes ampliacions al sistema que es pretenen incorporar a mig plaç:

- Ampliar el sistema d'accés per tal de permetre rols d'usuari. D'aquesta forma, es pot oferir accés al sistema a d'altres usuaris, limitant la informació a la que poden accedir.

  Per exemple, es pot oferir al gerent d'una sala accés als diferents informes només de la seva sala.

- Oferir la configuració remota de diferents paràmetres del joc, com el percentatge de pagament de la bola extra.

- Integrar aquest sistema en un altre projecte de l'empresa que permet la instal·lació remota de jocs en màquines (l'anomenat *Server based gaming*), que es gestionaria mitjançant aquesta aplicació.

- Afegir l'enviament automàtic d'incidències en cas de detecció per part d'una màquina, incloent el log i l'estat dels dispositius físics.

- Fer servir aquest sistema pel control de les màquines de joc tipus B a Espanya (les clàssiques dels bars).

Parlant de la vessant tècnica del projecte, personalment he quedat molt satisfet tant de la metodologia Scrum con del *framework* Ruby on Rails. Des de la primera especificació del projecte fins a la seva conclusió hi han hagut força canvis. La facilitat de desenvolupament i el ràpid prototipatge que ofereix Rails, units a la flexibilitat de la metodologia Scrum, han fet que des dels moments inicials del projecte el client hagi pogut veure una versió incompleta però funcional de l'aplicació i hagi així pogut adaptar els requeriments inicials a una versió més propera a les veritables necessitats. Voldria a més destacar la importància de les proves unitàries i funcionals. Gràcies a aquestes s'han pogut afegir molts canvis sense por a cometre errors.

Per finalitzar, m'agradaria comentar la meva experiència amb Ruby on Rails. Cal dir que la meva experiència professional ha estat allunyada del desenvolupament web, així que aquesta era la meva primera presa de contacte amb un *framework* web. Havia sentit a parlar molt bé de Ruby on Rails i un dels principals motius per fer aquest projecte era poder provar aquesta tecnologia. He de dir que la feina realitzada ha estat molt satisfactòria, Rails és molt agraït en el sentit en que és molt senzill fer que les coses funcionin. El concepte *convenció sobre configuració*, una de les claus de Rails, fa que tot simplement funcioni, alliberant al desenvolupador de gran part de la feina més feixuga i permetent que es pugui centrar en l'aplicació en sí. Un cop acostumat a la seves particularitats, tot sembla força natural i lògic.

10. Bibliografia

Llibres

- RUBY, Sam; THOMAS, Dave; HEINEMEIER, David: *Agile Web Development with Rails*. *Third Edition*. 2009. 784p. ISBN 9781934356166

Pàgines web

[1] "November 2010 Web Server Survey". Netcraft, 28 de novembre del 2010.

[2] Web Services Architecture, W3C Working Group Note, 11 de febrer del 2004.

[3] Architectural Styles and the Design of Network-based Software Architectures (capítol 5), Roy T. Fielding, 2000.

[4] www.rubyonrails.org

[5] Ruby on Rails: All Aboard the Fast Train to Web Application Development, developer.com

[6] Rails 1.2: REST admiration, HTTP lovefest, and UTF-8 celebrations

[7] www.prototypejs.org

[8] script.aculo.us

[9] A Brief History of Ajax, Aaron Swartz, 2005.

[10] XMLHttpRequest, W3C

[11] Scrum.org

[12] Package: rails, paquet bàsic de Ruby on Rails

[13] Debian Lenny - Installing Passenger with Apache

[14] Package: libpgsql-ruby, ofereix connexió entre Ruby amb la base de dades PostgreSQL.

[15] Authentication, Authorization and Access Control, Apache HTTP Server Documentation