Universitat Autònoma
de Barcelona

SIMULADOR DE PRODUCTIVITAT EN UNA EXPLOTACIÓ LLETERA

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Gestió
realitzat per
Guillem Espinosa Martin
i dirigit per
Gonzalo Vera Rodríguez

Escola Universitària d'Informàtica
Sabadell, Juny de 2010

CERTIFICAT

El sotasignat, Gonzalo Vera Rodríguez,
professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria
ha estat realitzat sota la seva direcció
per en ***Guillem Espinosa Martin***

I per a que consti firma la present.
Sabadell, Juny de 2010

Signat: Gonzalo Vera Rodríguez

Universitat Autònoma de Barcelona

Document d'autorització per a introduir els Treballs dels alumnes a dipòsits digitals de la UAB i del CBUC

Nom i Cognoms de l'Autor: Guillem Espinosa Martin

DNI o Passaport: 44024516D

Com a únic titular dels drets de propietat intel·lectual del treball (títol):

SIMULADOR DE PRODUCTIVITAT EN UNA EXPLOTACIÓ LLETERA

Autoritzo a la Universitat Autònoma de Barcelona (UAB) i al Consorci de Biblioteques Universitàries de Catalunya (CBUC) a dipositar aquest treball al *Dipòsit de la Recerca de Catalunya (*RecerCat*)* o qualsevol altre creat per la UAB o el CBUC amb les finalitats de facilitar la preservació i la difusió de la recerca i la investigació universitària.

Per tant, autoritzo a la UAB, i al CBUC a realitzar els actes que siguin necessaris per tal d'introduir el treball als esmentats dipòsits, així com per preservar-lo i donar-li accés mitjançant comunicació pública. Aquestes institucions no estan obligades a reproduir el treball en els mateixos formats o resolucions en què serà dipositat originàriament. La cessió de l'exercici dels drets necessaris per tal de realitzar totes aquestes accions es fa amb caràcter de no exclusivitat, és a dir, sóc lliure de publicar-lo a qualsevol altre lloc.

Declaro que no vulnero cap dret de tercers ja sigui de propietat intel·lectual, industrial, secret comercial o qualsevol altre, en subscriure aquesta autorització, ni en relació al contingut d'aquest treball, de manera que exonero la UAB i el CBUC de qualsevol obligació o responsabilitat davant qualsevol acció legal que es pugui suscitar derivada del treball dipositat.

Finalment declaro que accepto que des del repositori es doni accés al treball mitjançant una llicència *Creative Commons*, "Reconeixement–NoComercial–SenseObraDerivada 2.5 Espanya" amb la qual es permet copiar, distribuir i comunicar públicament l'obra sempre que se'n citin l'autor original i la institució i no se'n faci cap ús comercial ni obra derivada.

Signatura

_____

Lloc i Data: Sabadell, 29 de Juny de 2010

RESUM

El projecte té com a objectiu el desenvolupament d'una aplicació de simulació del procés de productiu en una explotació lletera per dur a terme les classes pràctiques de l'assignatura de Producció Bovina a la Facultat de Veterinària de la Universitat Autònoma de Barcelona. Aquesta aplicació ha de ser una eina per a la formació de futurs professionals del sector. Es tracta doncs de desenvolupar un simulador d'ús amigable però amb la complexitat suficient perquè els usuaris puguin treure'n el màxim profit de l'experiència de simulació.

Aquest programari ha de permetre la interacció de diversos usuaris al mateix temps i l'accés a les dades generades a partir de les simulacions fetes amb els seus conjunts de vaques. També és necessari que els estudiants puguin consultar l'estat actual de les vaques amb l'objectiu de decidir de quina manera volen modificar els factors clau de la simulació per tal d'optimitzar la producció. Finalment s'ha de resoldre també el propi procés de la simulació amb una aplicació que alhora extregui dades de la base de dades i s'alimenti dels paràmetres d'entrada dels usuaris.

Per tant es requereix d'un conjunt de tecnologies que resolguin els tres aspectes principals: la comunicació de diversos usuaris amb el programari de manera simultània, la gestió de les dades de l'escenari de simulació i finalment la pròpia simulació.

Aquesta aplicació servirà per millorar qualitativament les classes de pràctiques de producció bovina i servirà per adaptar aquesta part de l'assignatura al nou marc formatiu europeu del Pla de Bolonya.

INDEX

1. INTRODUCCIÓ

1.1. MOTIVACIÓ

El projecte neix per la necessitat de desenvolupar una eina de formació en el camp de la producció lletera bovina i el seu promotor és el professor de la Facultat de Veterinària de la Universitat Autònoma de Barcelona, Sergi Calsamiglia.

La raó per la qual s'ha escollit aquesta proposta es fonamenta en que per dur-lo a terme és necessari aplicar el conjunt d'habilitats i de coneixements que s'ha d'adquirir durant l'any en àrees com la metodologia i gestió de projectes, la planificació de la producció, la enginyeria del software i les bases de dades. A més representa la oportunitat per aprendre a fer servir un llenguatge de programació orientat a objectes en el desenvolupament d'una aplicació.

En cas de dur a terme el projecte amb èxit, aquest desenvolupament representa una excel·lent experiència de cara a la incorporació al mon laboral a part de significar que s'han assolit les habilitats i els coneixements estudiats durant l'últim any.

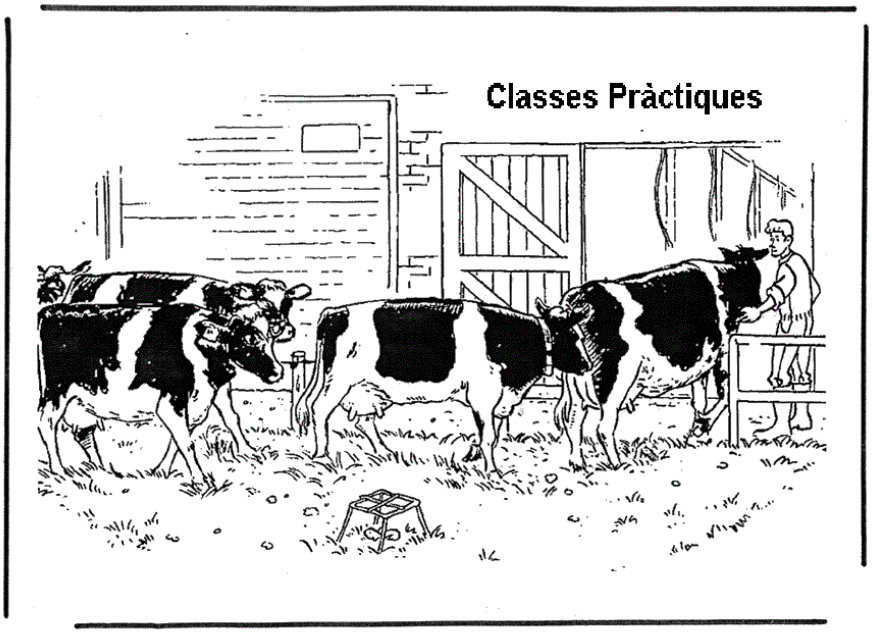


Fig. 1 - Cal resoldre el problema de fer pràctiques sense vaques

1.2. OBJECTIUS

El professor, a part de jugar el rol de client del projecte, també proporciona tota la informació tècnica en el camp de la producció bovina necessària per a confeccionar la simulació.

El desenvolupament del projecte compta de tres parts importants:

1:: **La comunicació de diversos usuaris de forma simultània amb el programari.** Cal cobrir des de les interfícies per administrar o fer ús del simulador fins a la gestió dels diversos fluxos de comunicació amb el simulador.

2:: **La gestió de les dades dels escenaris de simulació.** L' estat de cada una de les vaques associades a cada usuari i la configuració dels indicadors clau, tan del bestiar com de l'entorn, que incideixen en la productivitat dels animals.

3:: **La simulació que, combinant les instruccions dels usuaris i les dades de l'escenari, mostri el resultat de la simulació.** La simulació s'obté a partir de diversos mòduls que representen les diferents àrees d'influència en la productivitat d'una explotació lletera.

A partir d'aquesta primera diferenciació dels aspectes funcionals del programari es pot determinar quins objectius ha de perseguir el projecte:

- Determinar els perfils d'usuaris amb el seu conjunt d'interfícies.
- Permetre la comunicació simultània de varies aplicacions client, amb diferents ubicacions i amb una aplicació servidor que gestioni les peticions sense posar en risc la integritat de les dades dels usuaris.
- Determinar el modelat de dades que ens permeti anar transformant les dades durant les simulacions d'un usuari.
- Desenvolupar els mòduls de simulació que representin la interacció dels factors que poden influir en la productivitat lletera bovina.
- Combinar els factors d'entrada amb els mòduls per determinar els resultats de la simulació.

1.3. ESTRUCTURA DE LA MEMÒRIA

La memòria desenvolupa un estudi de la viabilitat del projecte. S'analitza quina són les necessitats que cal resoldre i de quina manera s'han de desenvolupar les solucions per determinar si es viable des del punt de vista tècnic i de recursos.

Després s'analitza en profunditat quines son les funcionalitats esperades de l'aplicació i sota quines condicions s'ha de desenvolupar una solució als objectius del projecte. En aquesta part també es fa una anàlisi de la tecnologia que hi ha disponible i les alternatives que es poden fer servir.

Un cop analitzat el problema es planteja un disseny de la solució. Aquest disseny té com a prerequisit una explicació sobre la tria de les eines que es fan servir pel desenvolupament. Després es detalla per blocs el disseny de cada part del desenvolupament.

Amb una planificació més acurada llesta i els dissenys de tot el que cal fer es descriu de quina manera s'ha implementat la solució. Es descriuen parts de l'aplicació que mereixen una atenció especial per la seva complexitat o per la seva importància per entendre cóm funciona la solució.

Un cop finalitzada la implementació es descriuen les proves a les que s'ha sotmès el resultat final i quina mena de problemes s'han detectat i si s'han resolt.

A la part final de la memòria s'analitza la planificació real i es valora l'experiència. En aquest mateix bloc es fa un seguit de recomanacions sobre possibles millores i futures ampliacions.

2. ESTUDI DE VIABILITAT

2.1. INTRODUCCIÓ

Abans de començar a treballar en una solució que doni resposta als objectius del projecte cal fer un anàlisi de la situació, determinar quins son els requeriments que ha de complir aquest solució i determinar si es tenen els recursos tècnics, econòmics i temporals per dur a terme el projecte.

En primera instància s'analitza la situació actual i si existeixen solucions que donin resposta a aquesta solució. En cas de que no sigui així es determinen els requisits que ha de complir el projecte i es proposa una solució.

A partir de la solució proposada es fa una estimació de la planificació necessària per dur-la a terme, un anàlisi dels riscos i una avaluació dels costos i els beneficis.

2.2. ESTUDI DE LA SITUACIÓ ACTUAL

Actualment l'assignatura de producció bovina s'imparteix a partir d'unes sessions de teoria i unes sessions de resolució de problemes. Els estudiants tenen una visió abstracte de procés productiu que es limita a càlculs predictius de la producció del bestiar.

De cara l'any vinent s'ha de distribuir la planificació del curs en relació a la nova norma educativa del Pla de Bolonya. Es vol formar a futurs professionals amb els coneixements necessaris per optimitzar la producció de llet de les vaques. Dins d'aquesta formació cal exercitar als estudiants amb sessions pràctiques. El problema és que no es poden fer aquestes sessions amb els animals. Per tant es busca una eina perquè els alumnes treballin amb les dades reals que s'extreuen de l'observació dels cicles productius del bestiar.

Gràcies a aquesta eina es vol millorar la formació:

- Estimulant el grau de participació de l'alumnat per enriquir el seu aprenentatge.

- Afegint complexitat a la resolució de problemes a l'aula.
- Avaluant en funció de resultats obtinguts a partir d'una experiència simulada.
- Aportant una visió més real de cara a la futura incorporació al mercat laboral dels estudiants.

Els usuaris de la eina son:

- Professor: Responsables de transmetre els coneixements, de plantejar els exercicis i de comprovar si estan bé.
- Alumnes: Responsables d'adquirir els coneixements i resoldre els problemes que els professors els plantegen.

Aquesta eina ha de permetre als alumnes gestionar els factors que incideixen en la productivitat de tota una explotació lletera amb un nombre determinat de vaques. En funció de les seves decisions ha de mostrar quina incidència tenen aquestes decisions en la granja.

Els objectius del projecte son:

- Simular la producció diària d'una vaca en funció d'un ampli ventall de paràmetres.
- Emmagatzemar totes les dades derivades de la simulació.
- Que cada alumne tingui un marc de treball diferent.
- Generar informes per als estudiants i els professors.
- Gestió d'usuaris amb diferents perfils i les seves dades associades.
- Gestió de cursos i variables d'entorn de simulació específics per cada curs.
- Gestionar la recuperació d'informació en cas de pèrdua, errors o averies.
- Desenvolupar un conjunt d'interfícies amigables i accessibles.
- Generar documentació d'ajuda per a cada tipus d'usuari.

Solucions existents al mercat

Existeixen diverses aplicacions orientades a la gestió de granges de vaques al mercat. S'ha fet un petit estudi per determinar si amb aquest programari es poden cobrir els objectius plantejats en el projecte:

Vaquitec (http://www.agritecsoft.com/sp/vaquitec/**)**

És una aplicació per fer el seguiment d'una explotació bovina que disposa de:

- Informes per a la seguiment del bestiar.
- Utilitats per fer la formulació de racions.
- Una eina de gestió econòmica: ingressos, despeses, vendes, comptes, gràfics, factures, resums.
- Diferents mètodes d'entrada de dades i compatibilitat amb una àmplia varietat d'ordinadors de ma i sistemes electrònics d'identificació.
- Dissenyat utilitzant els estàndards de la industria amb una estructura client / servidor per a la base de dades.
- Windows com a entorn de funcionament.

Té un cost per llicència (granja) de 230€ per la versió de fins a 500 caps de bestiar. Això suposa adquirir suficients llicencies per una aula. Amb aquesta eina no es possible l'accés remot dels estudiants. Tampoc permet fer simulacions, per tant algú hauria de generar dades per els possibles escenaris i que els alumnes determinessin quines accions durien a terme.

Taurusweb (http://www.taurusweb.com/**)**

És un planificador de recursos empresarials (ERP) orientat a explotacions bovines amb tota mena de interfícies amb indicadors clau per a tots els àmbits de la gestió d'una explotació bovina. Entre un ampli ventall d'eines destaquen:

- Generació de bases de dades a partir de les històries dels animals.
- Generació d'indicadors per a l'anàlisi, planificació i presa de decisions.

- Integració de bases de dades a Internet, en www.taurus webs.com, per generar estadístiques regionals, nacionals i internacionals.
- Entre molts altres indicadors, permet fer el seguiment de: població, animal, control d'entrades i sortides, reproducció, palpacions, parts, castracions, transferència d'embrions, producció de llet.
- Disposa de simuladors en la versió Prèmium

Té un cost de 5000€ per la versió prèmium que és la que disposa de simulador. L'aplicació és massa complexa i té moltes eines que no entren dins dels objectius de la formació i poden complicar excessivament el seu ús. Tampoc queda clar quina mena de llicencia o paquet de llicencies hauria d'adquirir la universitat per fer-ne ús.

Queda clar doncs que no hi ha aplicacions al mercat que cobreixin els objectius del projecte per que no es tracta d'un programari orientat a la formació. Està orientat al suport d'una activitat productiva comercial i per tant no disposa ni de les funcionalitats necessàries per fer el seguiment del progrés dels estudiants ni de la estructura per gestionar diferents granges. Per tant cal desenvolupar una aplicació que respongui als requeriments que satisfacin els objectius del projecte.

2.3. REQUERIMENTS DEL PROJECTE

Per definir els requeriments del projecte cal conèixer les funcionalitats que la solució ha de incloure. El professor és el patrocinador del projecte i és el responsable de definir aquestes funcionalitats aportant les dades necessàries per a definir el comportament esperat de la solució.

En funció de les característiques de l'entorn i les restriccions per l'ús que tindrà la solució s'ha de definir també uns requeriments no funcionals i unes restriccions.

Finalment hi ha una proposta de l'equip de treball necessari per dur a terme el projecte i s'explicita el material que s'ha de lliurar.

Requeriments Funcionals

Els requeriments funcionals son el recull de funcionalitats esperades de la solució. Aquestes funcionalitats descriuen de quina manera interactua l'usuari amb l'aplicació i quines utilitats li proporciona.

- Manteniment (altes, baixes, modificacions) de perfils d'usuaris de l'aplicació.
- Generació automàtica d'entorns de simulació per els alumnes.
- Control d'accés dels usuaris de l'aplicació.
- Gestió de les variables de simulació.
- Generació de llistats de l'estat del bestiar.
- Generació de gràfiques de progrés de producció de llet.
- Generació de llistats de progrés dels alumnes.
- Generació de canvis en el bestiar en funció de les simulacions.
- Generació i manteniment de l'històric productiu i reproductiu de llet de cada vaca.
- Gestió de canvis d'estat de les vaques.
- Manuals d'ajuda per als usuaris.
- Còpies de seguretat i recuperació de dades.

Requisits no funcionals

Els requeriments no funcionals descriuen altres requeriments necessaris per definir bé el que es necessita de l'aplicació. Aquest requeriments no estan relacionats amb les funcionalitats que l'aplicació ha de tenir però si que poden estar relacionades amb l'entorn en el que s'executa l'aplicació i les condicions en les que se'n farà ús.

- Interfície amigable usable per usuaris no experts.
- Control de totes les entrades d'usuaris.
- Protegir els resultats de simulació de modificacions per part dels alumnes.

- Generar entorns de simulació a partir de llistats d'alumnes en format fitxers de dades separades per comes (CSV).
- La seguretat de les dades: el servidor ha de tenir restriccions d'accés físic.

Restriccions del sistema

Les restriccions del sistema fan referència a condicions importants que s'han de tenir perquè d'alguna manera limiten les seves la tria de les possibles solucions als objectius del projecte.

- L'aplicació s'ha d'implementar amb independència del Sistema Operatiu.
- L'aplicació ha d'estar disponible a través d'Internet.
- L'aplicació s'ha d'adaptar al sistema físic disponible.
- El projecte ha d'estar finalitzat abans del 28 de juny de 2010.
- L'aplicació s'ha de desenvolupar en un llenguatge orientat a objectes.
- L'aplicació ha de permetre un accés concurrent de al voltant de 50 usuaris simultanis.

Equip del projecte

Nom	**Responsabilitat**
Cap del projecte	Defineix, gestiona, planifica i controla el projecte.
Analista	Col·labora amb el cap de projectes en l'estudi de viabilitat i la planificació. Analitza l'aplicació: arquitectura, metodologia, especificacions, estàndards,... Participa en el disseny.
Programador	Dissenya i desenvolupa l'aplicació d'acord amb l'anàlisi i planificació prevista. Participa en el procés de implantació.
Tècnic de proves	Participa en el disseny de les proves internes i externes. Realitza les proves i participa en el procés de control de qualitat.

Producte i documentació del projecte

- S'ha de lliurar una aplicació informàtica.
- S'ha d'elaborar uns manuals d'instal·lació i d'ús per als perfils d'usuari
- S'ha de lliurar una memòria del projecte.

2.4. PROPOSTA DE SOLUCIÓ I ALTERNATIVES

Es proposa desenvolupar una eina de formació que dóna resposta als objectius del projecte. Una eina que permet als estudiants simular una granja de vaques productores de llet i que permet als professors fer un seguiment de la simulació per avaluar el grau en que els alumnes han assolit els coneixements de l'assignatura.

Aquesta eina és una aplicació web que permet la interacció dels usuaris amb una granja virtual que s'emmagatzema en una base de dades. L'entorn web es familiar per als usuaris i facilita l'aprenentatge de l'aplicació. També és un model clàssic de l'arquitectura client servidor que permet la connexió de diverses aplicacions client (navegadors) amb un servidor que serveix les dades.

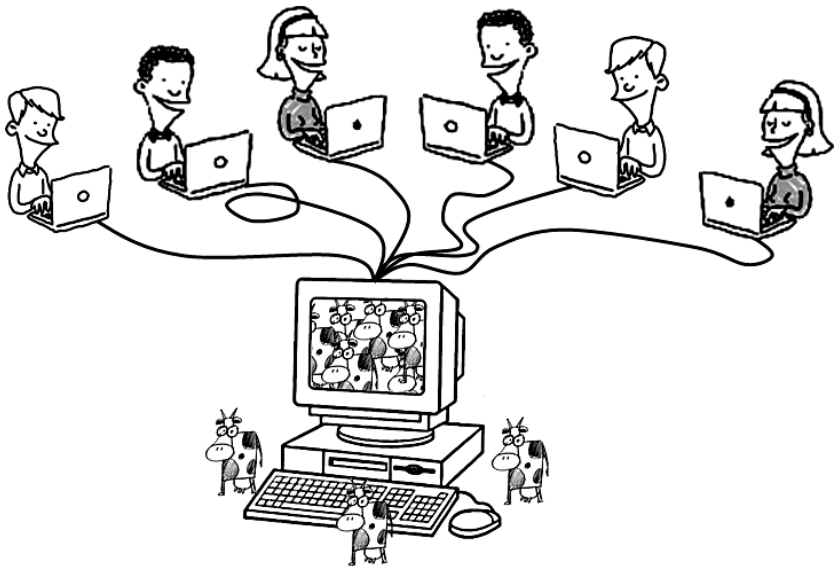


Fig. 2 - L'aplicació web facilita l'accés dels usuaris des de qualsevol lloc

Cal dissenyar una estructura de dades amb un llenguatge de programació orientat a objectes per imitar una granja de vaques real. També cal dissenyar un

conjunt d'interfícies web per permetre la interacció dels usuaris amb aquesta estructura fent ús d'una sèrie d'instruccions. Finalment cal dissenyar una base de dades per emmagatzemar el estat actual de la estructura de dades i les dades històriques generades pels usuaris en la seva interacció amb la granja.

Existeixen un seguit d'alternatives tecnològiques per dur a terme aquesta aplicació web que cal estudiar per determinar el conjunt d'aquestes que millor s'ajusta per implementar la solució proposada.

2.5. PLANIFICACIÓ DEL PROJECTE

El projecte es desenvolupa del 16 de novembre de 2009 al 8 de març de 2010 amb una dedicació mitjana de 25 hores setmanals. El total d'hores dedicades al projecte serà de 323,4 hores.

Recursos del projecte

Recursos humans	**Valoració**
Cap de projecte	100 €/h
Analista	50 €/h
Programador	30 €/h
Tècnic proves	20 €/h

Recursos materials: S'utilitzen els recursos materials disponibles a la universitat. Tot el desenvolupament es fa utilitzant programari de domini públic tret de la planificació i control.

Costos indirectes: amortització dels recursos de desenvolupament.

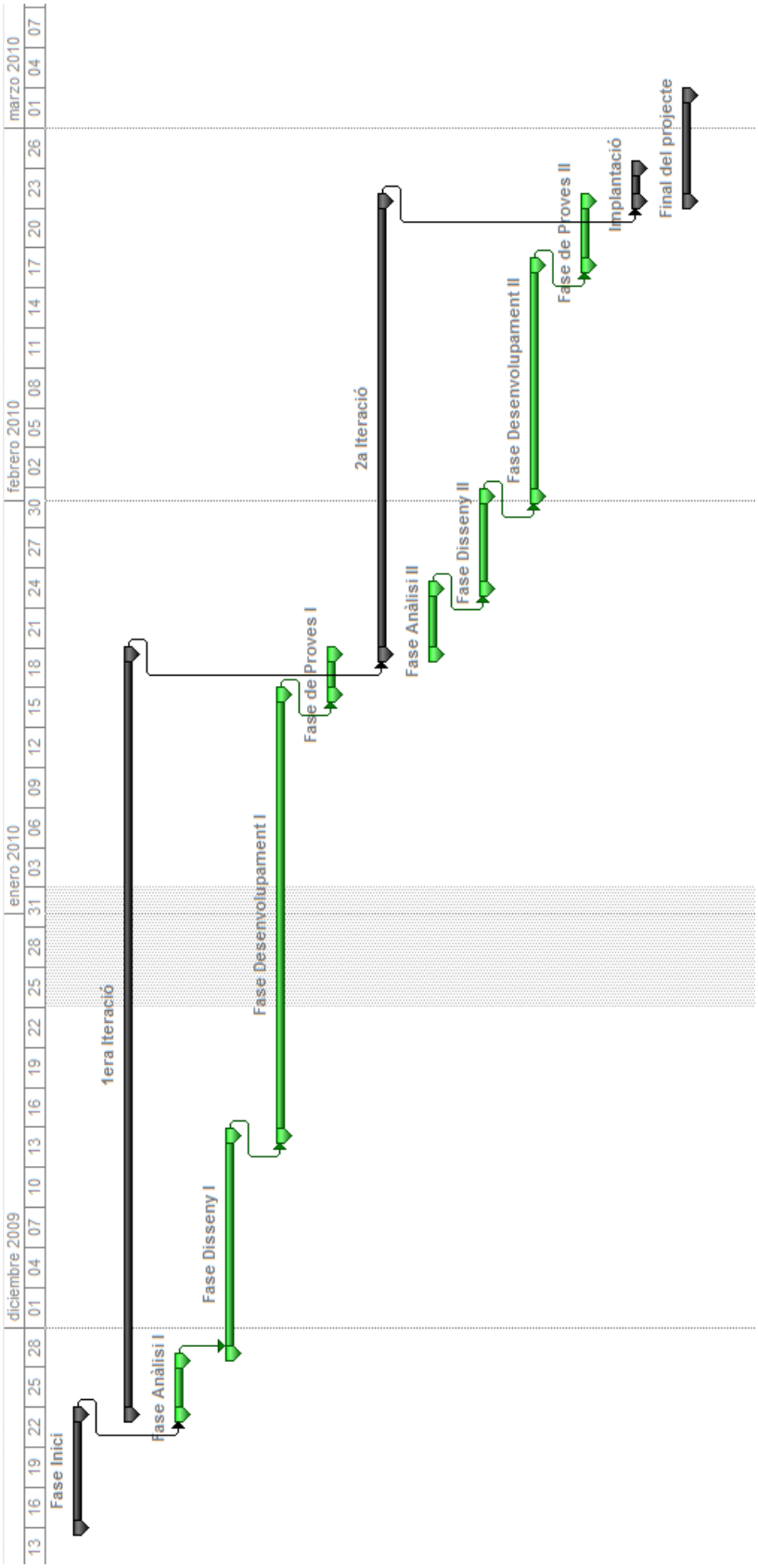
Tasques del projecte

Tot seguit es detallen les tasques que s'han planificat per dur a terme el projecte. S'ha fet sevir Microsoft Project com a eina de planificació i control.

Nom de la Tasca	Durada	Inici	Fi
Fase Inici	**3,13d**	**16/11/09**	**24/11/09**
Inici del projecte: assignació i matriculació del projecte	2h	16/11/09	16/11/09
Estudi de requisits des del punt de vista del usuari	2h	16/11/09	17/11/09
Estudi de viabilitat	20h	17/11/09	23/11/09
Aprovació estudi de viabilitat	1h	24/11/09	24/11/09
1era Iteració	**14,75d**	**24/11/09**	**20/01/10**
Fase Anàlisi I	**1,75d**	**24/11/09**	**28/11/09**
Anàlisi de requisits (casos d'ús)	6h	24/11/09	26/11/09

Anàlisi de dades (bases de dades)	4h	26/11/09	27/11/09
Documentació de l'anàlisi	3h	27/11/09	28/11/09
Aprovació de l'anàlisi	1h	28/11/09	28/11/09
Fase Disseny I	**3,88d**	**29/11/09**	**15/12/09**
Estudi de llenguatge de desenvolupament	8h	29/11/09	01/12/09
Disseny de la base de dades	5h	01/12/09	03/12/09
Disseny de l'aplicació	1d	03/12/09	09/12/09
Disseny de la interfície	4h	10/12/09	11/12/09
Disseny de les proves	3h	11/12/09	14/12/09
Documentació del disseny	2h	14/12/09	14/12/09
Aprovació de disseny	1h	15/12/09	15/12/09
Fase Desenvolupament I	**8d**	**15/12/09**	**17/01/10**
Preparació de l'entorn de desenvolupament	4h	15/12/09	16/12/09
Configuració de la base de dades	2h	16/12/09	17/12/09
Mòdul de comunicació client/servidor	4h	17/12/09	18/12/09
Mòdul recuperació i emmagatzematge base de dades	4h	18/12/09	21/12/09
Mòduls de simulació i generació de dades	50h	22/12/09	17/01/10
Fase de Proves I	**1,13d**	**17/01/10**	**20/01/10**
Proves unitàries I	1h	17/01/10	17/01/10
Proves d'integració I	2h	18/01/10	18/01/10
Proves d'estrès I	2h	18/01/10	19/01/10
Documentació de desenvolupament i test	3h	19/01/10	20/01/10
Aprovació de desenvolupament i test	1h	20/01/10	20/01/10
2a Iteració	**14,25d**	**20/01/10**	**23/02/10**
Fase Anàlisi II	**1,75d**	**20/01/10**	**25/01/10**
Revisió de requisits	2h	20/01/10	21/01/10
Anàlisi nous requisits	4h	21/01/10	22/01/10
Anàlisi de dades	2h	22/01/10	23/01/10
Anàlisi de seguretat i legalitat	5h	23/01/10	24/01/10
Documentació de l'anàlisi	1h	25/01/10	25/01/10
Fase Disseny II	**1,88d**	**25/01/10**	**01/02/10**
Revisió disseny de la base de dades	2h	25/01/10	25/01/10
Revisió disseny de l'aplicació	4h	26/01/10	27/01/10
Revisió disseny de la interfície	1h	27/01/10	27/01/10
Disseny de l'ajuda en línia	2h	27/01/10	28/01/10
Redisseny de les proves	3h	28/01/10	29/01/10
Revisió de la documentació del disseny	2h	29/01/10	29/01/10
Aprovació modificacions documentació	1h	01/02/10	01/02/10
Fase Desenvolupament II	**9d**	**01/02/10**	**18/02/10**
Reconfigurar base de dades	1h	01/02/10	01/02/10
Revisar mòdul comunicació	1h	01/02/10	01/02/10
Revisió mòdul de simulació i generació de dades	70h	02/02/10	18/02/10
Fase de Proves II	**1,63d**	**18/02/10**	**23/02/10**
Proves unitàries II	2h	18/02/10	19/02/10
Proves d'integració II	2h	19/02/10	19/02/10
Proves d'estrès II	4h	22/02/10	22/02/10
Revisió de documentació de desenvolupament i test	0,5d	22/02/10	23/02/10
Aprovació de canvis de desenvolupament i proves	1h	23/02/10	23/02/10
Implantació	**2,38d**	**23/02/10**	**25/02/10**
Instal·lació	5h	23/02/10	23/02/10
Proves reals	10h	24/02/10	25/02/10
Formació d'usuaris	4h	25/02/10	25/02/10
Final del projecte	**4,63d**	**23/02/10**	**03/03/10**
Generació de documentació (memòria)	30h	23/02/10	02/03/10
Tancament del projecte	2h	02/03/10	02/03/10
Defensa del projecte	5h	02/03/10	03/03/10
PROJECTE	**39,14d**	**16/11/09**	**03/03/10**

Planificació temporal



2.6. AVALUACIÓ DE RISCOS

Llista de riscos i pla de contingència

	RISC	PLA DE CONTINGÈNCIA
R.1	**Planificació temporal optimista:** No s'acaba en la data prevista i augmenten els recursos.	Ajornar alguna funcionalitat i afrontar possibles pèrdues.
R.2	**Manca alguna tasca necessària:** No es compleixen els objectius del projecte.	Revisar l'estudi de viabilitat i modificar la planificació.
R.3	**Pressupost poc ajustat:** Menys qualitat i/o pèrdues econòmiques.	Renegociar amb el client per afrontar possibles pèrdues.
R.4	**Canvi de requisits:** Endarreriment en els desenvolupament i el resultat.	Renegociar amb el client per ajornar funcionalitats i/o modificar planificació.
R.5	**Disseny de la solució inadequat:** Endarreriment en la finalització del projecte i pèrdua de qualitat.	Millorar la formació de l'equip i preveure dissenys alternatius.
R.6	**Eines de desenvolupament inadequades:** Endarreriment en la finalització del projecte amb una pèrdua de qualitat i/o la impossibilitat de complir algun dels objectius del projecte.	Millorar la formació de l'equip, preveure eines alternatives i millorar la qualitat.
R.7	**No es fa correctament la fase de test:** Manca de qualitat i deficiències en l'operativa que provoca la insatisfacció dels usuaris i una pèrdua econòmica.	Dissenyar els test amb temps, realitzar tests automàtics, donar garanties, afrontar pèrdues econòmiques.
R.8	**Abandonament del projecte abans de la finalització:** Pèrdues econòmiques i frustració.	No té solució.
R.9	**Manca d'implantació de mesures de seguretat:** Pèrdua o manipulació d'informació, no desitjada que provoca la insatisfacció dels usuaris i una pèrdua econòmica.	Revisar la seguretat en cada fase, aplicar polítiques de seguretat actives.
R.10	**Dificultats per accedir al client:** Manquen requisits o són inadequats. Això provoca endarreriments i la insatisfacció dels usuaris.	Fixar un calendari de reunions per millorar el contacte amb el client.

2.7. AVALUACIÓ DE COSTOS I BENEFICIS

Estimació cost de personal

Recursos humans	**Hores**	**Cost**
Cap de projecte (CP)	65,4	6540 €
Analista (A)	60,6	3030 €
Programador (P)	181,35	5440,50 €
Tècnic proves (TP)	16,05	321 €

Estimació cost dels recursos

	Cost amortització	**Cost unitari**	**Període amortització**	**Període utilització**
Amortització PC Programador	100 €	1200 €	36 m.	3 m.
Amortització Microsoft Office	20,8 €	250 €	36 m.	3 m.
Amortització Microsoft Project	30 €	360 €	36 m.	3 m.
TOTAL	**150,8 €**			

Estimació del cost de les activitats

Tasca	Costos	Recursos
Fase inici	2.500,00 €	
Inici del projecte: assignació i matriculació	200,00 €	CP
Estudi de requisits des del punt de vista del usuari	200,00 €	CP
Estudi de viabilitat	2.000,00 €	CP
Aprovació del estudi de viabilitat	100,00 €	CP
1era iteració	4.314,00 €	
Fase anàlisi I	**725,00 €**	
Anàlisi de requisits (casos d'ús)	300,00 €	A
Anàlisi de dades (bases de dades)	200,00 €	A
Documentació de l'anàlisi	150,00 €	A
Aprovació de l'anàlisi	75,00 €	A[50%] CP[50%]
Fase disseny I	**1.321,50 €**	
Estudi dels llenguatges de desenvolupament	272,00 €	A[20%] P[80%]
Disseny de la base de dades	200,00 €	A[50%] P[50%]
Disseny de l'aplicació	320,00 €	A[50%] P[50%]
Disseny de la interfície	260,00 €	A P[50%]
Disseny de les proves	94,50 €	A[50%] P[25%] TP[25%]
Documentació del disseny	100,00 €	A

Tasca	Costos	Recursos
Aprovació de disseny	75,00 €	A[50%] CP[50%]
Fase desenvolupament I	**1.920,00 €**	
Preparació de l'entorn de desenvolupament	120,00 €	P
Configuració de la base de dades	60,00 €	P
Mòdul de comunicació client/servidor	120,00 €	P
Mòdul recuperació i emmagatzematge de dades	120,00 €	P
Mòduls de simulació i generació de dades	1.500,00 €	P
Fase de proves I	**347,50 €**	
Proves unitàries I	25,00 €	TP P[50%]
Proves d'integració I	70,00 €	TP P[50%]
Proves d'estres I	100,00 €	TP P
Documentació de desenvolupament i test	90,00 €	P
Aprovació de desenvolupament i test	62,50 €	CP[50%] A[25%] P[25%]
2a iteració	4.037,50 €	
Fase anàlisi II	**750,00 €**	
Revisió de requisits	150,00 €	CP[50%] A[50%]
Anàlisi nous requisits	200,00 €	A
Anàlisi de dades	100,00 €	A
Anàlisi de seguretat i legalitat	250,00 €	A
Documentació de l'anàlisi	50,00 €	A
Fase Disseny II	**547,50 €**	
Revisió disseny de la base de dades	80,00 €	A[50%] P[50%]
Revisió disseny de l'aplicació	160,00 €	A[50%] P[50%]
Revisió disseny de la interfície	40,00 €	P[50%] A[50%]
Disseny de l'ajuda en línia	0,00 €	
Re disseny de les proves	97,50 €	A[25%] P[50%] TP[25%]
Revisió de la documentació del disseny	100,00 €	A
Aprovació modificacions documentació	70,00 €	A[25%] P[25%] CP[50%]
Fase desenvolupament II	**2.160,00 €**	
Reconfigurar base de dades	30,00 €	P
Revisar mòdul comunicació	30,00 €	P
Revisió mòdul de simulació i generació de dades	2.100,00 €	P
Fase de proves II	**580,00 €**	
Proves unitàries II	100,00 €	P TP
Proves d'integració II	100,00 €	P TP
Proves d'estres II	200,00 €	P TP
Revisió de documentació de desenvolupament	120,00 €	P
Aprovació de canvis de desenvolupament i proves	60,00 €	CP[40%] A,P,TP ->[20%]
Implantació	780,00 €	
Instal·lació	220,00 €	A[70%] P[30%]
Proves reals	360,00 €	P[40%] TP[20%] A[40%]
Formació d'usuaris	200,00 €	A
Generació de documentació (memòria)	3.000,00 €	CP
Tancament del projecte	200,00 €	CP
Defensa del projecte	500,00 €	CP

Resum i anàlisi cost benefici

Cost de desenvolupament del projecte	*15331,50 €*
Cost d'amortització del material	*150,80 €*
Total:	***15482,30 €***

Actualment és calcula que els estudiants paguen amb les seves matrícules, de mitjana, el 10% del cost real del seus estudis a la universitat pública. Si es té en compte que el cost de matriculació a l'assignatura de producció bovina a la facultat de veterinària de la Universitat Autònoma de Barcelona té un cost estimat per a l'alumne de 160 € (mitja de cost per repetir convocatòria) i la part del temari dedicat a la producció de llet és un terç de l'assignatura. Aleshores per a la universitat, el les pràctiques de producció bovina d'un alumne tenen un cost estimat en:

Cost de les pràctiques d'un alumne = ((160€ x 100/10) – 160€) / 3 = 480 €

A més amb el canvi que suposa el Pla Bolonya s'han de programar més hores d'aprenentatge de l'alumne per el mateix preu que s'està cobrant ara.

És calcula que amb aquesta aplicació es podrà fer l'equivalent al doble d'hores de formació que amb la solució actual a classe per què els exercicis son més directes i és pot seguir formant als alumnes amb feina que es podrà fer amb l'aplicació des de casa seva. Per tant amb aquesta solució s'aconsegueix un estalvi de 480 € per alumne a l'any en concepte d'hores de formació.

Amb una mitjana de 20 estudiants matriculats a l'assignatura cada any el programari es pot amortitzar en 22 mesos amb un marge de 2400€.

2.8. CONCLUSIONS

Des del punt de vista tècnic s'ha determinat que no es pot resoldre els objectius del projecte amb programari existent en el mercat i que cal desenvolupar una aplicació web a mida de les especificacions del client. És viable tècnicament ja que existeix un ventall d'eines que s'ajusten a les necessitats de implementació i es disposa de la formació necessària durant l'any acadèmic per afrontar les tasques necessàries per assolir els objectius del projecte.

A partir d'una planificació temporal s'ha determinat que es disposa del temps necessari per entregar el programari a mida de les necessitats del client a temps per a la seva utilització en la formació de l'any acadèmic vinent.

També s'ha analitzat la seva viabilitat a nivell de recursos davant la situació d'adaptar els estudis al Pla Bolonya que dispara el cost per a la Facultat de Veterinària en concepte d'hores de formació. Amb una inversió inicial important que es pot amortitzar en 22 mesos s'aconsegueix resoldre el problema que representa la situació actual. Tot plegat fa que es determini que el projecte és viable.

3. ANÀLISI

3.1. INTRODUCCIÓ

En aquesta part de la memòria es presenta un anàlisi del problema que es vol resoldre. És presenta una sèrie d'especificacions funcionals i no funcionals que l'aplicació ha de tenir. S'analitzen els requeriments del programari des dels més genèrics (la visió externa de l'aplicació i el comportament de la simulació) fins als detalls més específics (el comportament de la vaca i el seu encaix dins de la granja). També es defineixen els requeriments que no responen a les funcionalitats que el programari ha de tenir però si que son necessaris per assolir-los. Tenen a veure amb factors com el rendiment esperat o l'entorn en el que s'ha de treballar.

Dins d'aquest anàlisi també s'estudien les opcions que ens ofereix el marc tecnològic per resoldre aquests requeriments i tenir arguments per valorar quin conjunt d'eines es vol fer servir per dur a terme el projecte.

Finalment s'estudien les diferents arquitectures de software tradicionals per veure de quina manera organitzarem la distribució de funcionalitats que a de recollir el programari a desenvolupar.

3.2. ESPECIFICACIONS FUNCIONALS

Per descriure les funcionalitats que ha d'incloure l'aplicació es descriuen els requeriments que han de permetre la seva configuració, el seu manteniment i la gestió dels usuaris que hi tenen accés. Després es descriuen les funcionalitats bàsiques per controlar l'accés i les eines a disposició dels usuaris. Finalment es descriu la funcionalitat concreta de simular i de quina manera s'ha de comportar la simulació definint el comportament de les vaques i les granges.

Administració i gestió de l'aplicació

Hi ha una seguit de funcionalitats que son necessàries per gestionar l'aplicació i que no han de ser accessibles per als alumnes. Aquest conjunt de funcionalitats

es pot separar en dos blocs en funció de si son eines d'administració i configuració de l'aplicació o eines per gestionar els grups d'estudiants.

Les funcionalitats per administrar i configurar l'aplicació son:

- Configuració d'accés dels professors a l'aplicació.
- Afegir, modificar o eliminar cursos i afegir o treure professors dels cursos.
- Modificar les variables d'entorn de simulació que es carreguen per defecte per cada nou curs.
- Fer copies de seguretat manuals de la base de dades de l'aplicació.

Les funcionalitats que permeten la gestió dels grups d'alumnes que fan servir l'aplicació han de permetre fer el seguiment dels alumnes i resoldre problemes puntuals que aquest puguin tenir. Aquestes funcionalitats inclouen:

- Modificar les variables de l'entorn de simulació d'un curs per conduir les simulacions d'un grup d'alumnes.
- Afegir alumnes a un curs, ja sigui d'un en un o a partir d'un llistat CSV que generi els perfils dels alumnes.
- Modificar les dades personals dels alumne i les dades d'accés a l'aplicació. També cal l'opció d'esborrar un alumne i la seva granja.
- Definir un sistema perquè els usuaris puguin generar la seva granja dins d'un curs.
- Mostrar llistats que comparin els indicadors del estat de cada una de les granges d'un curs.
- Donar accés a la granges del curs per poder fer un seguiment personalitzat dels alumnes més exhaustiu.
- Exportar el llistat de les granges del curs amb els seus indicadors en format CSV per facilitar-ne l'avaluació amb qualsevol programari de full de càlcul.

Ús de l'aplicació

El conjunt de funcionalitats que han d'estar disponibles per als alumnes ha d'incloure:

- Permetre accés a la granja de cada usuari mitjançant un nom d'usuari i una paraula de pas.
- Mostrar dades amb indicadors clau de l'estat de la granja.
- Mostrar gràfiques de punts construïdes a partir de la producció de totes les vaques en funció del dia de lactància i del nombre de parts.
- Exportar les dades de les vaques d'una granja en format CSV que permeti manipular-les amb un programari de full de càlcul.
- Mostrar taula de seguiment de la fertilitat de les vaques.
- Mostrar les dades relacionades amb cada una de les vaques. Aquestes dades inclouen gràfics amb l'evolució en la producció de llet i un històric de successos rellevants per la vaca.
- Posar a disposició dels estudiants un conjunt d'accions per poder influir sobre l'estat de les vaques: inseminar, assecar, sacrificar i descobrir en quin estat es troben.
- Permetre la compra de nous animals.
- Permetre simular el dia a dia de la granja.

La simulació, la granja i les vaques

Per definir de quina manera es du a terme la simulació cal especificar el comportament de les vaques i de quina manera es relacionen amb la granja. La granja serveix per agrupar les vaques dels usuaris i per recollir les dades generades per els animals. També és útil per mantenir la informació relacionada amb l'estat de la simulació de cada usuari. Informació com el nombre de dies simulats per l'alumne o el número de vaques adquirides o sacrificades. Per tant quan es simula un dia de la granja es simula un dia de totes les vaques de la granja. Tota la informació generada per un dia de simulació s'ha de guardar en per comparar el progrés de la granja.

Per simular un dia de la vaca cal conèixer de quina manera es comporta una vaca. S'ha dibuixat la xarxa de petri que descriu els canvis d'estat de la vaca per fer un model del seu comportament i de quins factors fan que canviï el seu estat (Fig. 3). En l'aplicació les vaques parteixen d'un estat en que acaben de parir per primera vegada, per tant no hi ha vaques que no siguin potencialment productores de llet. La vaca tan bon punt a parit comença a produir llet i en segueix produint fins que se la asseca (més endavant es defineix el comportament d'aquesta producció de llet). Al cap d'uns pocs dies comença de nou el cicle menstrual interromput durant la gestació. Un cop comença aquest cicle es va repetint fins que es torna a quedar prenyada. Perquè això passi, cal inseminar-la durant un període de temps molt curt al final del cicle que és quan la vaca està en zel. Si durant aquest petit espai de temps la vaca és inseminada té un cert percentatge de probabilitats de quedar prenyada i no se sap si ho està fins que torna a passar un cicle menstrual sencer. Tot i que un observador extern no sap el seu estat fins al cap d'uns dies, la vaca ja pot estar prenyada i, si és així, ho estarà durant un llarg període de temps després del qual parirà i tornarà a donar llet. Torna a produir llet perquè la pràctica habitual és assecar la vaca perquè no doni més llet uns quants dies abans del part. Si no es fa la seva producció futura de llet és veu afectada negativament.

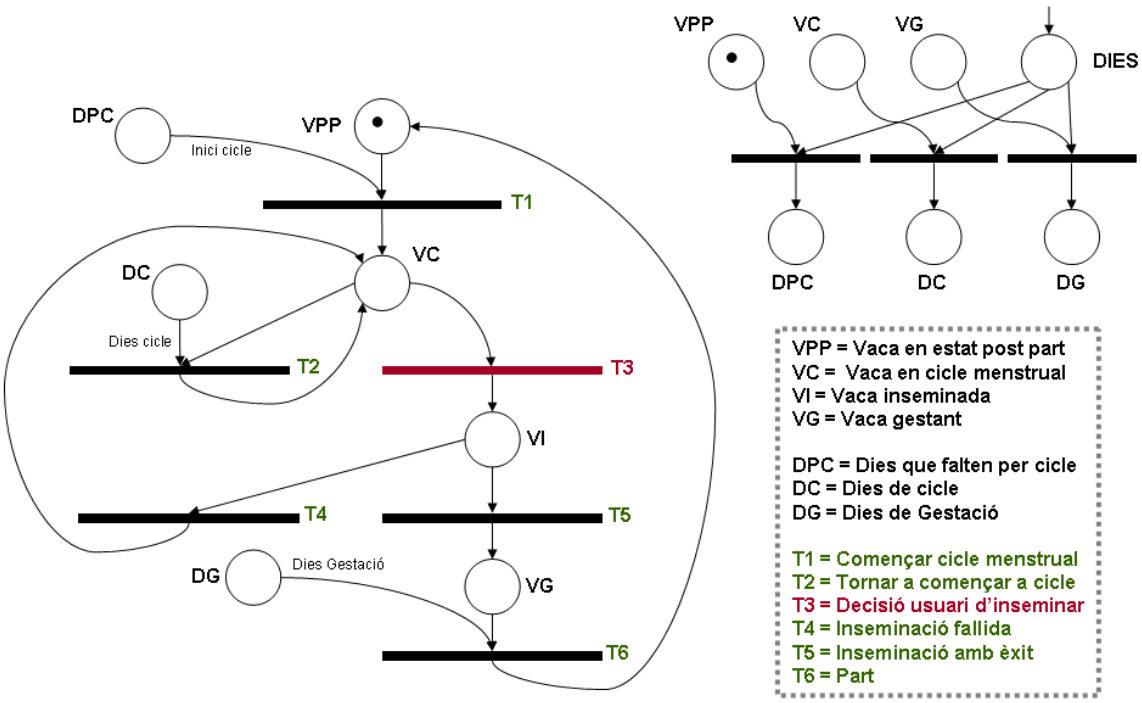


Fig. 3 - Xarxa de Petri del model de simulació dels estats de la vaca

A part del comportament de la vaca el professor de la facultat de veterinària ha especificat amb equacions i constants de quina manera es comporta la producció d'una sola vaca i quines son les variables que s'han de tenir en compte. A partir d'aquestes es genera la següent llista de d'especificacions referents a la vaca:

- Una vaca esta sempre en un estat de 4 possibles: en el impàs entre el part i el primer cicle menstrual, durant el cicle menstrual, en període de gestació i, si l'alumne la sacrifica, morta. No hi ha vaques abans del seu primer part a l'aplicació.
- Una vaca morta no participa de les simulacions i per tant només està present per consultar-ne la historia.
- Una vaca en estat després del part comença a produir llet i controla si han passat prou dies per començar el cicle menstrual.
- Una vaca està repetint el cicle menstrual mentre no passa a estar prenyada. El canvi a estat prenyada només és possible si és inseminada quan està en zel.
- Mentre la vaca està gestant controla els dies que falten pel part. Si s'asseca a la vaca uns dies abans del part la seva producció futura es manté en els mateixos nivells, si no es fa la producció a la següent lactància baixa.
- El moment òptim per inseminar la vaca és quan està en zel. Per aquest dia té una probabilitat de quedar prenyada que és menor si es inseminada el dia anterior o següent.
- No es pot saber si la inseminació ha estat un èxit fins que ha passat un cicle menstrual sencer i uns quants dies especificats més de marge. Un cop passat aquest temps es pot descobrir l'estat de la vaca.
- Es pot inseminar i assecar o sacrificar la vaca en qualsevol moment.
- S'ha de guardar tota la història productiva, de fertilitat i d'assecat de la vaca.
- S'ha de guardar variables de nivells de greix, proteïna i cèl·lules somàtiques de la vaca.

- S'ha de poder ajustar la simulació canviant els possibles valors de les variables que determinen la producció, la fertilitat i l'estat de la vaca per tal de poder aconseguir entorns de simulació específics.
- La producció de llet de la vaca es determina a partir dels dies de lactància que porta la vaca, el nombre de parts i els càlculs aleatoris que determinen el càlcul de producció anual potencial i els períodes de pujada de producció, de pic de producció i de baixada de producció. També s'aplica una petita variabilitat a la producció diària per donar més realisme a la simulació.
- La forma de determinar la producció és en funció d'uns paràmetres que els professors poden modificar per alterar la producció en les següents fórmules es mostren els valors que inicialment s'han definit però poden canviar per apropar la simulació al comportament real de les vaques:

Producció anual potencial	=	Aleatori entre [7000,11000]
Pic de producció	=	Per un part: 0,75 x Producció anual potencial /200 Per més d'un part: Producció anual potencial /200
Dies de pujada de producció	=	Per un part: aleatori entre [60-80] Per més d'un part: aleatori entre [35-55]
Dies de durada del pic de producció	=	Per un part: Aleatori entre [25-35] Per més d'un part: Aleatori entre [15-25]
Producció del primer dia de lactància	=	0.7 x Pic de producció

- La producció d'un dia *n* es determina en funció del següent quadre:

	Dies en lactància < Dies de pujada producció	Dies de pujada producció < Dies en lactància < (Dies de pujada producció + Dies de durada del pic de producció)	(Dies de pujada producció + Dies de durada del pic de producció) < Dies en lactància
Vaca amb un part	1,009 × Producció del dia n-1	1 × Producció del dia n-1	0,998 × Producció del dia n-1
Vaca amb més d'un part	1,016 × Producció del dia n-1	1 × Producció del dia n-1	0,9916 × Producció del dia n-1

3.3. ESPECIFICACIONS NO FUNCIONALS

Hi ha també un seguit d'especificacions relacionades amb l'entorn en el que ha de funcionar l'aplicació i les característiques necessàries per complir els seus objectius:

- Permetre accés concurrent d'un grup d'usuaris i que aquest accés estigui disponible a través d'Internet.
- Fer ús d'interfícies amigables fàcils d'utilitzar per usuaris que no necessariament han de ser experts en l'ús d'eines informàtiques.
- Protegir les dades de l'aplicació prevenint la seva manipulació com a conseqüència d'un mal ús.
- Controlar l'accés a l'aplicació per garantir la independència dels exercicis de simulació de cada alumne.
- Fer ús de programari multi plataforma per facilitar-ne la seva utilització en l'àmbit acadèmic.

3.4. MARC TECNOLÒGIC

Per determinar com s'ha de dur a terme la implementació s'ha fet un anàlisi del marc tecnològic. Al analitzar les diferents eines de que es pot disposar per dur a terme l'aplicació s'ha tingut en compte que es planteja fer servir una arquitectura de tres capes (Fig. 4). S'ha de permetre l'accés concurrent de diferents usuaris a una aplicació servidora que serveix les dades i fa els càlculs de la simulació. També s'ha tingut en compte la necessitat de poder accedir a l'aplicació de forma remota.

Es valora també la possibilitat de disposar de programació orientada a objectes. Aquest tipus de programació és molt útil quan es tracta de desenvolupar entitats ben definides formades per una sèrie d'atributs (que determinen el seu estat) i de mètodes (que permeten a aquesta entitat comunicar-se amb d'altres entitats). De fet no es casualitat aquest tipus de programació tingui el seu origen en un llenguatge dissenyat per fer simulacions. Les simulacions necessiten de les entitats per descriure l'entorn que es vol simular.

Llenguatges de programació

Fer servir un entorn web sembla doncs una excel·lent opció per permetre l'accés remot a l'aplicació. Hi ha un ampli ventall d'opcions per el desenvolupament d'aplicacions web.

ASP.NET és una eina de l'empresa Microsoft destinada al desenvolupament d'aplicacions web. ASP.NET detecta senzilles instruccions barrejades amb les etiquetes HTML i interpreta les instruccions a dur a terme. Això facilita molt el seu ús però al mateix temps en limita molt les possibilitats en el cas de necessitar un llenguatge orientat a objectes. Un altre dels problemes que planteja és que necessita un servidor amb un sistema operatiu de Microsoft. L'aplicació que serveix de servidor de les peticions web es el Internet Information Services que està inclòs en les darreres versions dels sistema operatius de Microsoft.

PHP és la solució per desenvolupar aplicacions web més popular. Aquesta és una eina similar a ASP en el sentit que també és un conjunt d'instruccions que es poden inserir entre etiquetes HTML, està molt orientat al treball amb Bases de Dades però a diferència de ASP és compatible amb la major part de sistemes operatius. Aquestes característiques son les que han convertit a PHP en la eina de generació de pàgines web dinàmiques més utilitzada. Tot i així, com en el cas de ASP, la seva facilitat complica el desenvolupament d'aplicacions més sofisticades i que requereixin una programació orientada a objectes tot i que en les versions més recents d'aquest llenguatge es permet una certa orientació a objectes. L'aplicació que es faria servir per atendre les peticions web es el Apache, un projecte de codi obert que pràcticament és un estàndard en servidors de pagines web.

Finalment està la opció de fer servir el llenguatge de programació Java que disposa d'una eina de desenvolupament d'aplicacions de servidor més complexa de fer servir que les anteriors però alhora molt més potent. Aquesta eina son els Servlets i permeten aprofitar tota la potencia d'un llenguatge de programació de més baix nivell que ASP o PHP com és el Java. Els Servlets son petites aplicacions que atenen peticions a través del navegador i generen contingut de qualsevol

tipus com a resposta. Java es basa en la programació orientada a objectes. La seva fortalesa és també la seva debilitat ja que és considerablement més complex desenvolupar serveis web en Java. En aquest cas es faria servir una branca de desenvolupament del projecte Apache que s'anomena Apache Tomcat per atendre les peticions al servidor. Aquest programari està específicament dissenyat per treballar amb aquest tipus de tecnologia.

Sistema gestor de la base de dades

No només cal una eina per implementar l'aplicació, també cal una base de dades que emmagatzemi tota la informació que es genera durant les simulacions i que servirà als alumnes per comprovar el seu progrés i als seus professors per avaluar el seu aprenentatge.

En aquest camp s'han analitzat dues opcions gratuïtes. La primera és PostgreSQL. Una solució amb un llenguatge propi anomenat PL/PgSQL que és molt similar al PL/SQL que fan servir les bases de dades Oracle. És la solució més propera als gestors de bases de dades comercials como pot ser el propi Oracle amb multitud d'opcions i utilitats.

L'altra opció és MySQL. A diferència de PostgreSQL és més ràpida a l'hora de fer operacions però té menys utilitats i característiques especials. El seu consum és més baix tot i no ser una bona opció per a grans volums de dades. MySQL és molt robust, tot i ser gratuït, i existeixen molts projecte interessants al voltant d'aquesta base de dades gràcies a la seva popularitat.

3.5. ARQUITECTURA DE L'APLICACIÓ

Tot i que en els requeriments es parla d'un model client servidor s'ha descartat l'arquitectura client servidor per que en aquest cas no és prou precisa per explicar la estructura del projecte. Aquest model es basa en un conjunt de funcionalitats en uns "clients" què interaccionen amb un conjunt de funcionalitats que s'allotgen e una única aplicació "servidor".

Per estructurar d'una manera més clara l'arquitectura de l'aplicació és més adequat que es basi en el model de programari de tres nivells. Aquest tipus

d'estructura es defineix per la separació de les funcions de l'aplicació en una capa per a la presentació de la informació (interfícies d'usuari), una altra per al càlcul o la lògica de l'aplicació (on es troba el conjunt d'instruccions) i una per a les dades. (Fig. 4).

La capa de presentació inclou tot el conjunt de funcions que tenen a veure amb la interacció del usuari amb l'aplicació. D'aquesta manera es separa el disseny de l'aspecte del programari dels requeriments funcionals que aquest pugui tenir.

La capa de càlcul o de lògica de l'aplicació inclou totes les funcionalitats que es recullen en els requeriments. El seu objectiu es agrupar tots el càlculs que es fan a partir dels inputs dels usuaris combinats amb les dades del sistema.

La capa de dades es la que inclou la definició de les entitats del sistema i s'ocupa de la persistència de les dades.

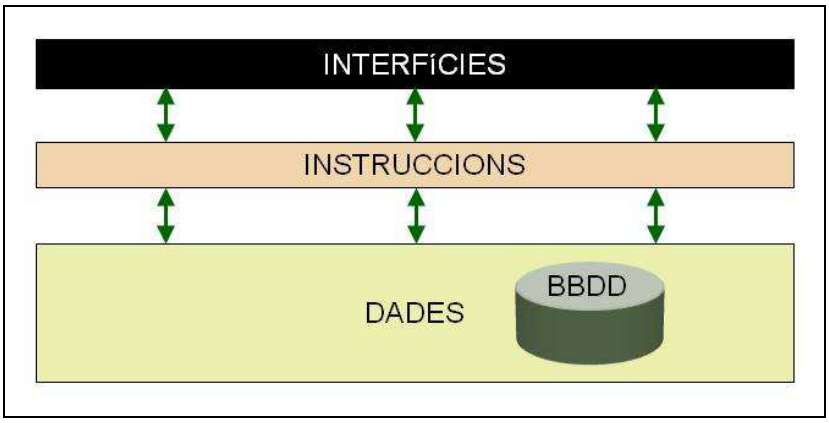


Fig. 4 – Arquitectura bàsica de l'aplicació

4. DISSENY

4.1. INTRODUCCIÓ

En aquest capítol es realitza el disseny de l'aplicació web. Aquest disseny es fa d'acord amb l'arquitectura de l'aplicació que es vol fer servir. Aquesta arquitectura de 3 nivells ha de permetre fer una clara diferenciació entre les interfícies de l'aplicació, les instruccions de l'aplicació que donen resposta a les funcionalitats que aquest ha de tenir i el modelat de les dades que permet mantenir la informació generada per l'ús de l'aplicació. Dins de cada capa de l'aplicació s'explica quines eines es vol fer servir i de quina manera aquestes eines han d'ajudar a desenvolupar l'aplicació.

A partir de l'anàlisi que s'ha fet dels requeriments s'ha decidit que cal distingir quines funcionalitats han d'estar disponibles segons l'usuari que accedeix a l'aplicació i per tant el primer és definir quins tipus d'usuari ha de tenir l'aplicació.

Després de dissenyar les capes de l'arquitectura de l'aplicació s'ha fet una nova planificació més acurada que recull amb més detall les tasques que es deriven del disseny de l'aplicació.

4.2. PERFILS D'USUARI

Després de fer l'anàlisi dels requeriments funcionals sembla evident que cal distingir les funcionalitats a les que tindrà accés un alumne de les que cal que tingui accés un professor. També es creu convenient separar les funcionalitats que permeten la configuració i el manteniment de l'aplicació de les funcionalitats que son necessàries per gestionar els alumnes i els cursos. No cal què els professors tinguin més opcions de les que els son necessàries per dur a terme les seves funcions. A partir d'aquest raonament s'han definit aquests perfils d'usuari en funció de les seves responsabilitats dins de l'aplicació:

Perfils d'usuari	Responsabilitat
Administrador del Sistema	▪ Gestió de comptes d'usuari de perfil professor. ▪ Gestió de cursos. ▪ Gestió de les variables per defecte de l'entorn de simulació de l'aplicació. ▪ Gestió de les copies de seguretat.
Professors	▪ Gestió comptes d'usuari de perfil alumne. ▪ Gestió de les variables d'entorn de simulació dels cursos.
Alumnes	▪ Fer el seguiment de la producció de llet. ▪ Prendre decisions sobre les vaques per controlar i millorar la producció. ▪ Aplicar les decisions simulant el dia a dia de la granja.

4.3. LA CAPA D'INTERFÍCIES

La capa d'interfícies ha de mostrar una estructura visual que permeti als usuaris disposar de totes les funcionalitats que ha de tenir l'aplicació. S'ha buscat una interfície que aprofités la simplicitat d'una pàgina web però afegint recursos que l'apropessin al comportament d'una aplicació tradicional (Fig. 5).

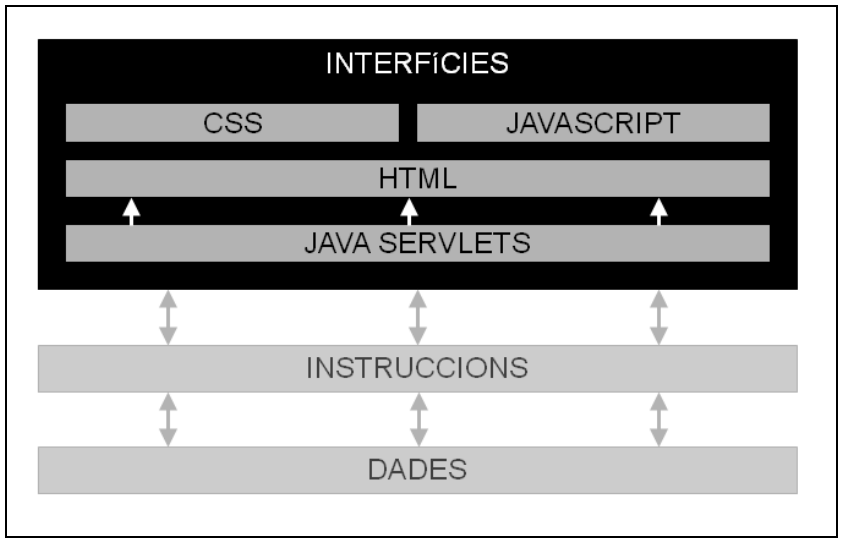


Fig. 5 - Les solucions que s'han fet servir a la capa d'interfícies

HTML, AJAX, CSS

A l'hora de decidir amb quines eines es desenvolupa l'aspecte de l'aplicació no s'han contemplat alternatives perquè les eines triades ja son les més àmpliament utilitzades i això mateix és el que es necessita per tenir una eina familiar per als usuaris. Un dels requeriments no funcionals de l'aplicació és que sigui amigable. Una interfície HTML és molt propera al que pot estar acostumat un estudiant avui dia i en aquest sentit és perfecte per escurçar el temps d'aprenentatge de l'aplicació per part dels alumnes. Però encara pot ser més amigable gràcies a la carrega dinàmica de continguts. Recarregant parts de la interfície sense carregar de nou tota la pàgina HTML que la defineix amb un marc tecnològic conegut com a Asynchronous Javascript And XML (Javascript asíncron i XML, AJAX) (Nieto, 2005).

AJAX és una tècnica de desenvolupament web per crear aplicacions interactives que s'executen en el client. S'afegeixen instruccions complexes a una pàgina web perquè el navegador dels usuaris mantingui una comunicació asíncrona amb el servidor en segon pla. D'aquesta manera és possible fer canvis sobre la pròpia pàgina sense tenir que recarregar tot el contingut. Fent la pàgina web una aplicació molt més interactiva, rapida i usable. Per aconseguir que el navegador faci totes aquestes peticions asíncrones cal afegir funcionalitats a aquest programa. Aquestes funcionalitats s'afegeixen mitjançant el Javascript. Un llenguatge que permet fer peticions asíncrones i manipular el contingut d'una pàgina web carregada pel navegador amb les respostes d'aquestes peticions (Zakas, y otros, 2006).

Aquest dinamisme es pot aconseguir amb una altra eina com és Flash. Però tot i que el seu llenguatge de programació, l'ActionScript 3.0, a evolucionat cap a Java i és relativament familiar per als programadors; presenta problemes de d'estàndards i es corre el risc de fer interfícies molt espectaculars però poc amigables per l'usuari.

També és important que es separi l'estructura de les interfícies d'usuari del seu disseny per tal de facilitar futures ampliacions o canvis visuals. La manera de fer-ho és fent ús de pagines d'estils (CSS). Dins d'aquest CSS es pot definir

l'aspecte de cada element etiquetat en HTML. De manera que es pot separar completament l'aspecte dels elements de una interfície de la seva estructura.

Tot i que aquestes estructures es basen en un entorn web i per tant han d'estar escrites en HTML es necessària una eina que escrigui aquest llenguatge HTML en funció dels continguts que es generen a l'aplicació.

Java Servlets i Apache Tomcat

Per a la gestió dels continguts visibles a la interfície de l'aplicació s'ha triat el llenguatge de programació Java i els seus Servlets. Aquets Servlets estan disponibles com a serveis web i l'estructura que permet això és el servidor web Apache Tomcat. Aquest programari suporta l'arquitectura dels Servlets i associa aquestes classes amb serveis disponibles per a peticions web. Aquets serveis admeten les peticions web GET i POST i generen una resposta en format HTML que és rep al navegador dels usuaris com una pàgina web. En el cas de les interfícies es fan servir les peticions GET perquè no s'ha controlar el pas de paràmetres o perquè aquests paràmetres no cal amagar-los al usuari (Chopra, 2004).

Un cop s'arrenca el servidor aquest inicia com a serveis tots el Servlets que estiguin definits un arxiu de configuració (Fig. 6), es carreguen en memòria les classes corresponents i a partir d'aquest moment per cada petició que rep algun d'aquets serveis és crea un fil d'execució. Aquests serveis web generen una resposta i aquesta resposta és la informació que veurà l'usuari en el seu navegador.

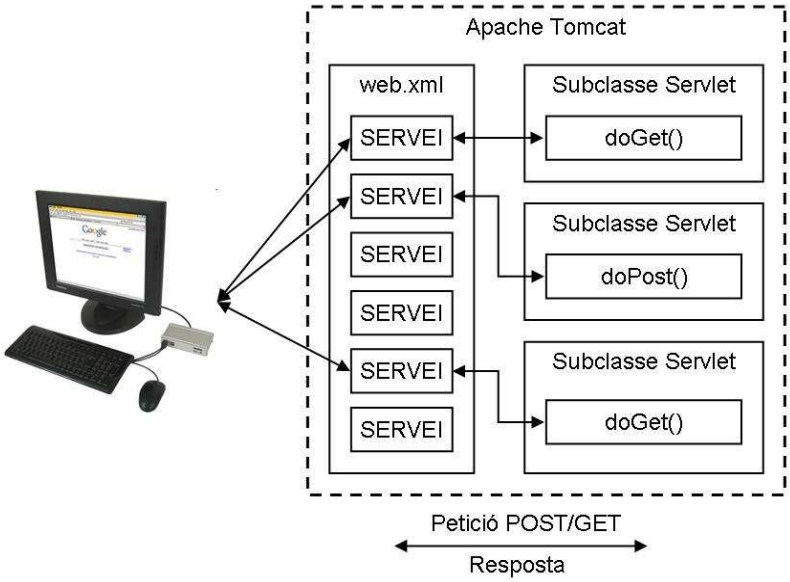


Fig. 6 - Com funcionen Apache Tomcat i els Servlets

El primer punt fort d'aquesta solució és que els Servlets son una classe de Java de la que s'hereta el mètode que rep la petició. De manera que el programador no s'ha de preocupar del nivell de comunicació entre client i servidor.

També disposa de solucions molt útils com son les Sessions que estan implementades en el propi paquet dels Servlets. Aquestes Sessions permeten a la aplicació web guardar informació a la sessió d'un usuari de l'aplicació per fer-ne ús més endavant. Això simplifica el control de les Sessions d'usuari i evita el tenir que estar passant contínuament informació d'una interfície a l'altra.

També es vol fer servir un paquet de classes ja implementades anomenades JFreeChart que resol la funcionalitat de generar gràfiques d'una forma molt més espectacular del que s'hagués pogut aconseguir implementant una classe sencera per poder mostrar aquest tipus d'informació visual.

Disseny de la interfície

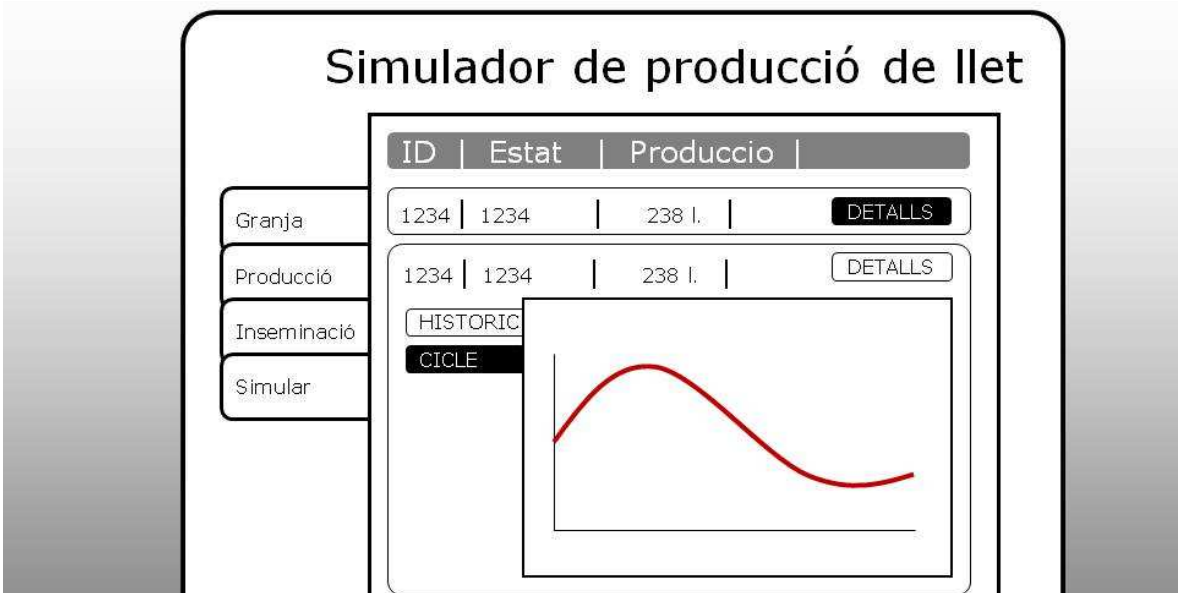


Fig. 7 - Disseny de la interfície

La interfície té un menú lateral per poder accedir a les diferents funcionalitats i un cos central on es carreguen les funcionalitats a les que té accés l'usuari (Fig. 7). Per accedir als estats de les vaques es fa servir un llistat amb files que tenen funcionalitats afegides amb Javascript. Aquestes funcionalitats es carreguen desplegant-se en funció de si es selecciona la fila. Com es mostra en el disseny dins d'aquestes funcionalitats està la de mostrar la gràfica de l'evolució en la producció. La idea és doncs de fer servir desplegables que carregin el contingut dinàmicament quan es vol mostrar el detall d'un element de la interfície. Per tant hi ha Servlets que carreguen només contingut parcial.

Les interfícies a més depenen de l'usuari i com que en funció de l'usuari tenen continguts comuns es defineix una jerarquia de classes que va encapsulat les funcionalitats que les subclasses necessiten (Fig. 8). L'arrel de la jerarquia conté mètodes per escriure la estructura bàsica del document HTML i els continguts comuns per totes les interfícies com la capçalera o el peu. A més es defineixen dos classes que agrupin també els mètodes per escriure els menús laterals en funció del tipus d'usuari i per gestionar elements dels llistats que té cada perfil d'usuari: en el cas dels alumnes els llistats de vaques i en el cas de l'administrador i els professors els llistats de granges.

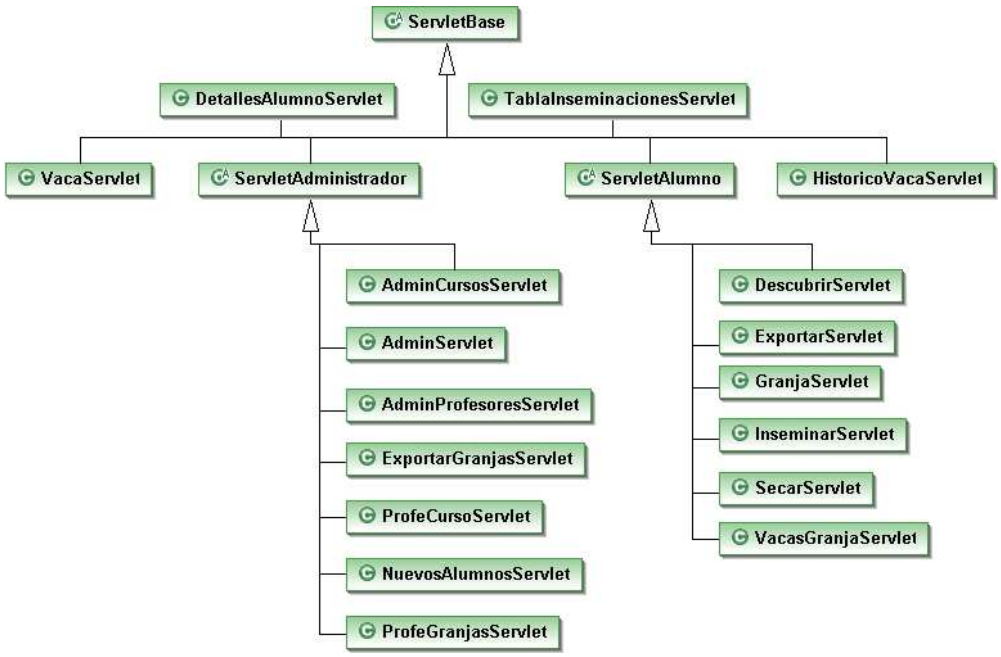


Fig. 8 - Diagrama de classes de la interfície

Per exemple a la Fig. 9, quan es carrega una interfície amb el llistat de les vaques d'una granja en realitat s'ha cridat un mètode de la classe VacasGranjaServlet que rep una petició web i com a resposta escriu codi HTML. La estructura bàsica del document HTML (Capçalera i Peu) s'escriu gràcies a un mètode de la classe arrel de la jerarquia (ServletBase), el menú lateral s'escriu amb un mètode de la seva classe mare ServletAlumno i fins i tot cada fila d'aquest llistat s'escriu amb un mètode de la pròpia classe ServletAlumno de manera que no es repeteixin parts de la interfície a cada classe d'aquesta jerarquia.

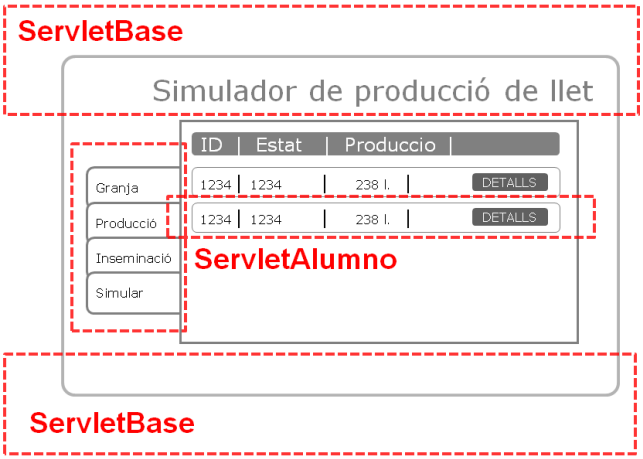


Fig. 9 - Exemple de l'aprofitament de la jerarquia de classes a la capa d'interfícies

4.4. LA CAPA D'INSTRUCCIONS

La capa d'instruccions és la capa que dona funcionalitat a les interfícies. A la interfície es dibuixa un desplegable o un botó que després està associat a funcionalitats que s'executen en aquesta capa. Aquestes funcionalitats han de treballar a més amb les estructures de la capa de dades per generar la resposta a les accions dels usuaris a la capa d'interfícies.

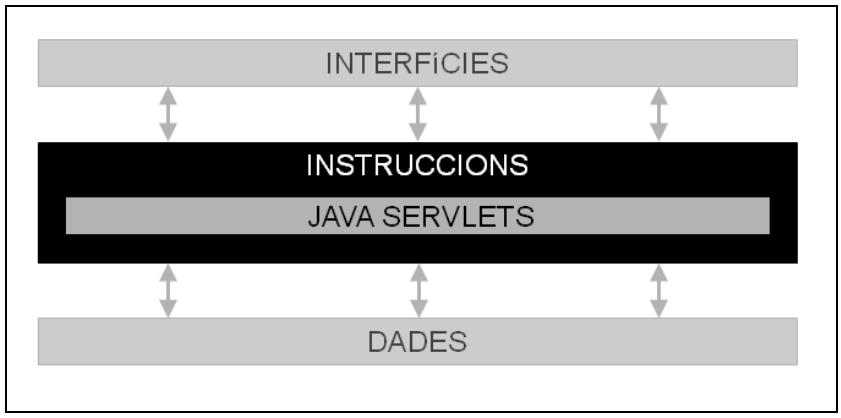


Fig. 10 - Les solucions que s'han fet servir a la capa d'instruccions

Per dur a terme aquestes tasques es torna a fer us dels Servlets (Fig. 10). Però ara aquets Servlets actuen de forma diferent. La seva resposta a les peticions no retorna HTML per carregar parts de la interfície. Aquets Servlets executen instruccions que modifiquen les dades, com per exemple generar una nova granja, i un cop executada la instrucció redirigeixen la resposta per que sigui una petició a una Interfície que és el que finalment es mostra a l'usuari. El seu funcionament queda amagat a l'usuari dotant a l'aplicació de les funcionalitats especificades en els requeriments funcionals.

Per exemple quan un usuari seleccionar un idioma de la interfície es crida la classe IdiomaServlet que guarda a la sessió de l'usuari la nova configuració i fa una petició al Servlet d'interfície des de on s'ha seleccionat aquest idioma perquè torni a generar el contingut amb el idioma escollit.

L'excepció son els Servlets que exporten dades, aquests retornen contingut en forma de fitxers que l'usuari pot guardar en disc. Però no afecten a la navegació de l'aplicació i de cara a l'usuari es com si es descarregués un fitxer allotjat al servidor.

La jerarquia de classes es basa també en el tipus d'usuari quan cal aprofitar mètodes (Fig. 11). En general son Servlets de poc codi i funcionalitat molt concreta. Els que implementen funcionalitats de simular instancien els objectes definits en la capa de dades i criden els mètodes d'aquets objectes.

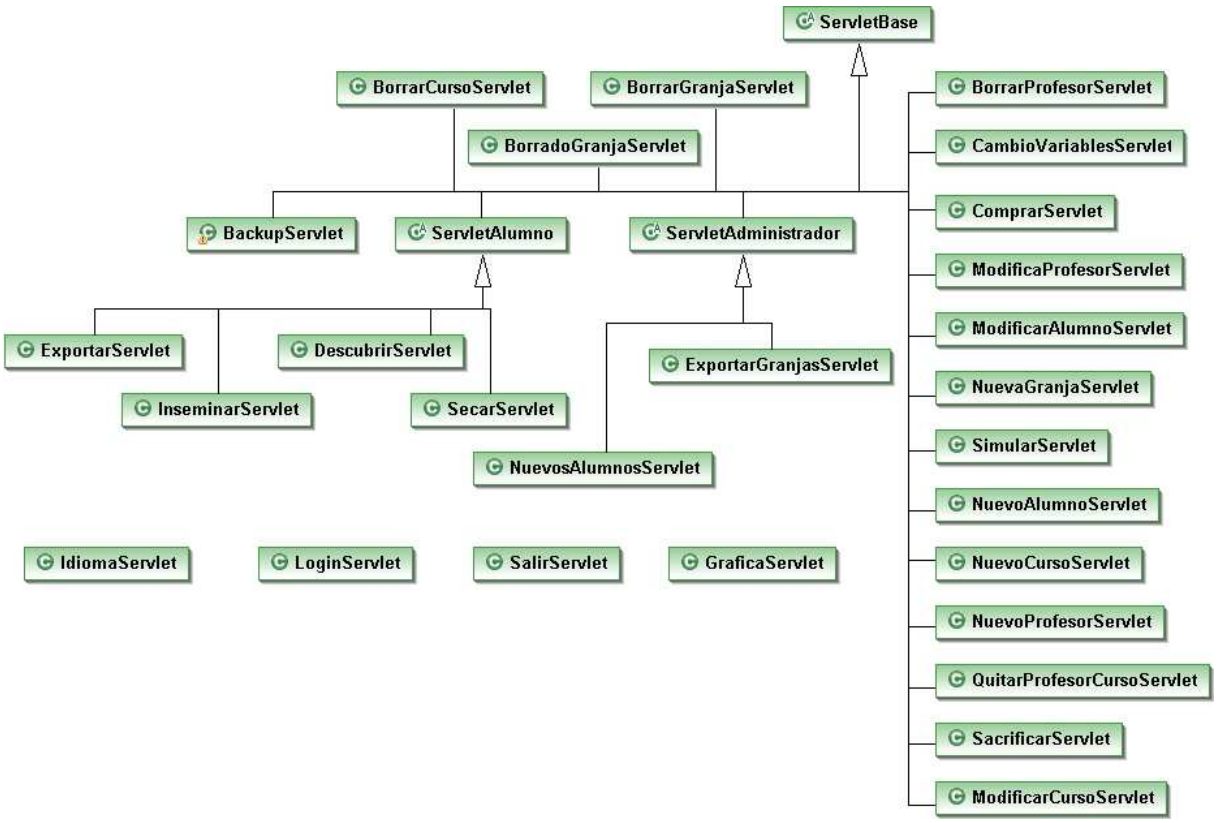


Fig. 11 - Diagrama de classes de la capa d'instruccions

4.5. LA CAPA DE DADES

Com s'ha vist a l'anàlisi existeixen unes entitats que son les que han de participar de la simulació. Aquestes entitats son estructures complexes que necessiten un conjunt d'atributs per determinar el seu estat i que necessiten de mètodes per comunicar-se amb les altres entitats. La capa de dades es on aquestes entitats es defineixen i l'eina que s'ha triat per desenvolupar-les és el Java (Fig. 12).

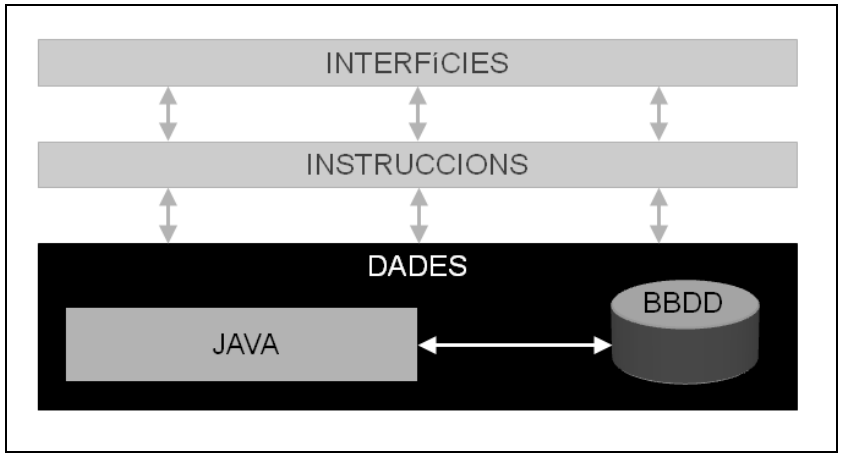


Fig. 12 - Les solucions que s'han fet servir a la capa de dades

Les altres capes ja tenen dissenys basats en arquitectures de Java però és en aquest nivell on aquest llenguatge és més adequat per la seva orientació a objectes. El Java permet crear classes que descriuen **entitats** que després es poden instanciar des de qualsevol altra classe de Java. Aquestes **entitats** poden ser independents de la implementació de les classes amb que es relacionen gràcies als mètodes amb que se les pot dotar. A més existeix el concepte de constructor que és un mètode especial que es crida cada cop que es crea un objecte de la **entitat** (Eckel, 2002).

Per exemple es defineix una classe vaca amb un conjunt d'atributs que determinen el seu estat com els dies de gestació o la seva edat. A aquesta classe li afegim un constructor que genera una vaca nova i inicialitza tots els seus atributs per que comenci amb un estat inicial després del primer part. Finalment podem afegir un mètode produir que determini quanta llet produeix en el estat actual. Aleshores des de qualsevol classe de Java puc instanciar una vaca i fer que aquesta vaca produeixi llet.

A la simulació tenim varies **entitats**, unes son bàsiques per entendre com funciona la simulació i d'altres ajuden a simplificar la seva implementació:

L'entorn de simulació

L'entorn de simulació és la **entitat** que defineix els marges en els que es mou l'escenari de simulació. La simulació no es determinista, per a unes mateixes condicions pot donar resultats diferents que es mouen dins dels marges d'aquest

escenari de simulació. Aquests marges son en realitat més de 50 variables que pot ajustar el professor i que especifiquen des del interval de possibles duracions del cicle menstrual de la vaca fins a la mida de la granja en nombre de vaques. Aquesta entitat abstracte ha de disposar de tots els mètodes necessaris per generar els valors aleatoris que es necessiten durant la simulació i tenir mecanismes per carregar o guardar aquests marges de l'escenari de simulació segons convingui.

La entitat granja

S'ha definit la granja com el conjunt de vaques d'un usuari i com a conjunt de dades que defineixen el progrés de l'estudiant. Per agrupar aquestes dades associades al progrés de l'estudiant es defineix una classe EstadoGranja que permeti mantenir una correspondència amb les dades que es generen després de cada simulació a nivell de la granja i guardar aquestes dades a la base de dades. També ens ha de permetre el procés invers de manera que sigui senzill carregar l'estat de la granja a partir del constructor de l'objecte EstadoGranja.

La entitat vaca

La vaca ha de recollir el comportament definit en el diagrama d'estats (Fig. 13). Ha d'evolucionar en funció del temps i les decisions que pren l'estudiant i cal controlar aquest canvi d'estats amb variables com les del diagrama. Cal a més el conjunt de variables que defineixen l'estat de salut la vaca i que ajuden a decidir quanta llet es produeix. La vaca ha de tenir accés a les variables d'entorn perquè en la major part dels casos aquestes variables determinen els canvis que es produeixen en la vaca.

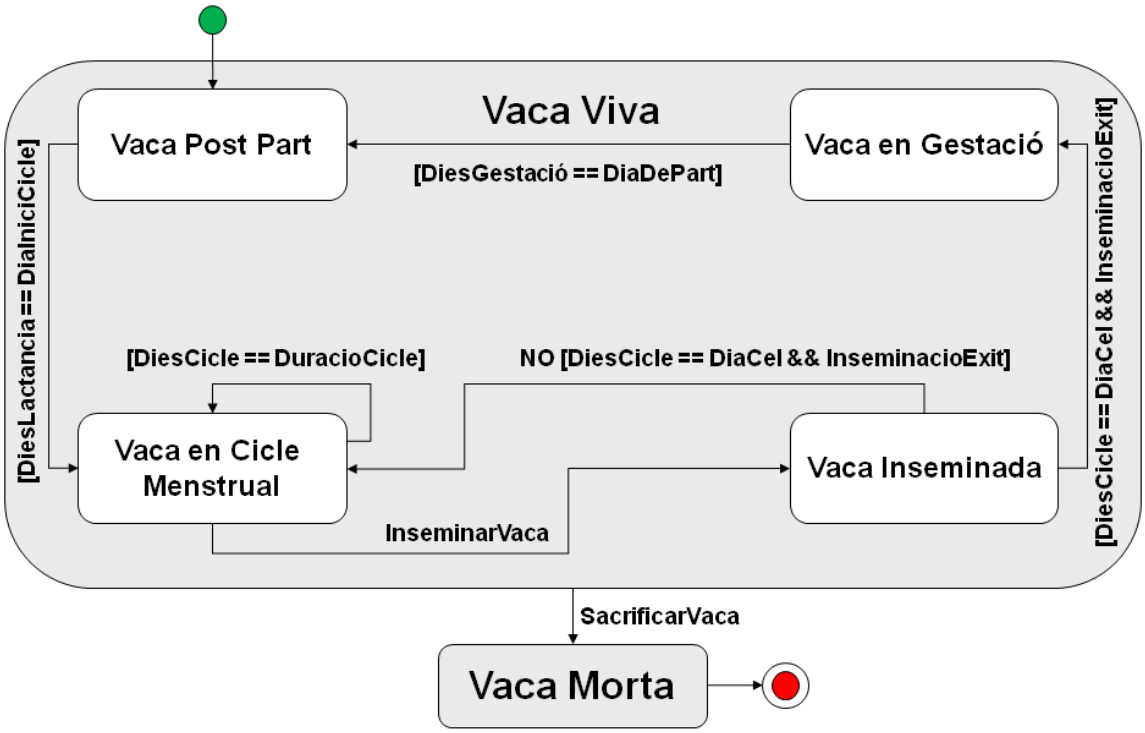


Fig. 13 - Diagrama d'estats de la vaca

Cal definir un mètode principal que gestioni els canvis d'estat tal com mostra el diagrama i que cridi d'altres mètodes en funció de l'estat en el que es troba. A partir d'aquest necessitarem els mètodes per:

- Produir llet: en funció de les especificacions.
- Inseminar: determina si ha tingut èxit i provoca el canvi d'estat.
- Secat: asseca la vaca i aquesta deixa de produir fins a un nou part.
- Descobrir estat: determina si la inseminació va tenir èxit per anotar-ho a l'històric i fer visible l'estat real de la vaca.
- Guardar estat: per que un cop s'hagin produït els canvis en la vaca aquets es guardin a la base de dades.

També s'han de definir constructors per generar vaques noves i carregar vaques a partir de la base de dades. És necessària una classe per guardar les decisions de l'estudiant sobre la vaca de manera que sigui senzill carregar les decisions de la base de dades o guardar-les si és necessari.

La base de dades

Per poder emmagatzemar les dades generades per l'aplicació així com els estats de les entitats que en formen part en cal una eina. Es fa servir MySQL per el seu baix consum de recursos i per ser més ràpid i àgil que PostgreSQL. Cal tenir present que volem que l'aplicació web sigui el més àgil possible per gestionar un accés concurrent d'usuaris fent consultes amb força volum de dades. Un volum de dades que no és prou gran com per que aquesta solució es quedi petita en front de l'altre sistema gestor de base de dades. Potser no té tantes funcionalitats com la seva competidora però al cap i a la fi son funcionalitats que tampoc es té pensat fer servir. És vol deixar a MySQL només el paper d'emmagatzemar dades per separar clarament la persistència de les dades dels càlculs que és fan amb aquestes dades.

La estructura de la base de dades ha de respondre al següent diagrama entitat – relació (Fig. 14). Per una banda tindrem com a mínim dos perfils d'usuari. Un tipus a pertany a curs com a granja (alumne) i l'altre ensenya durant el curs. La granja a més té associades un conjunt de vaques que s'hi estan.

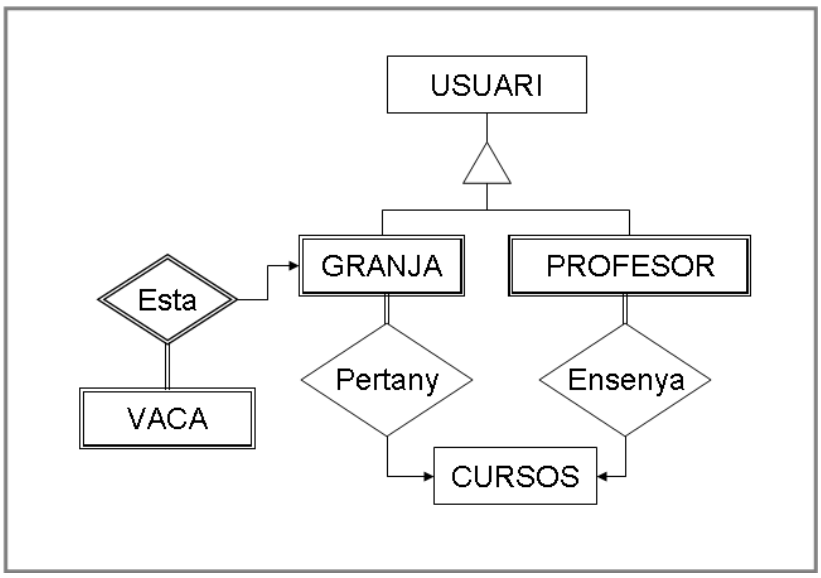


Fig. 14 - Diagrama entitat - relació de la base de dades

4.6. PLANIFICACIÓ TEMPORAL

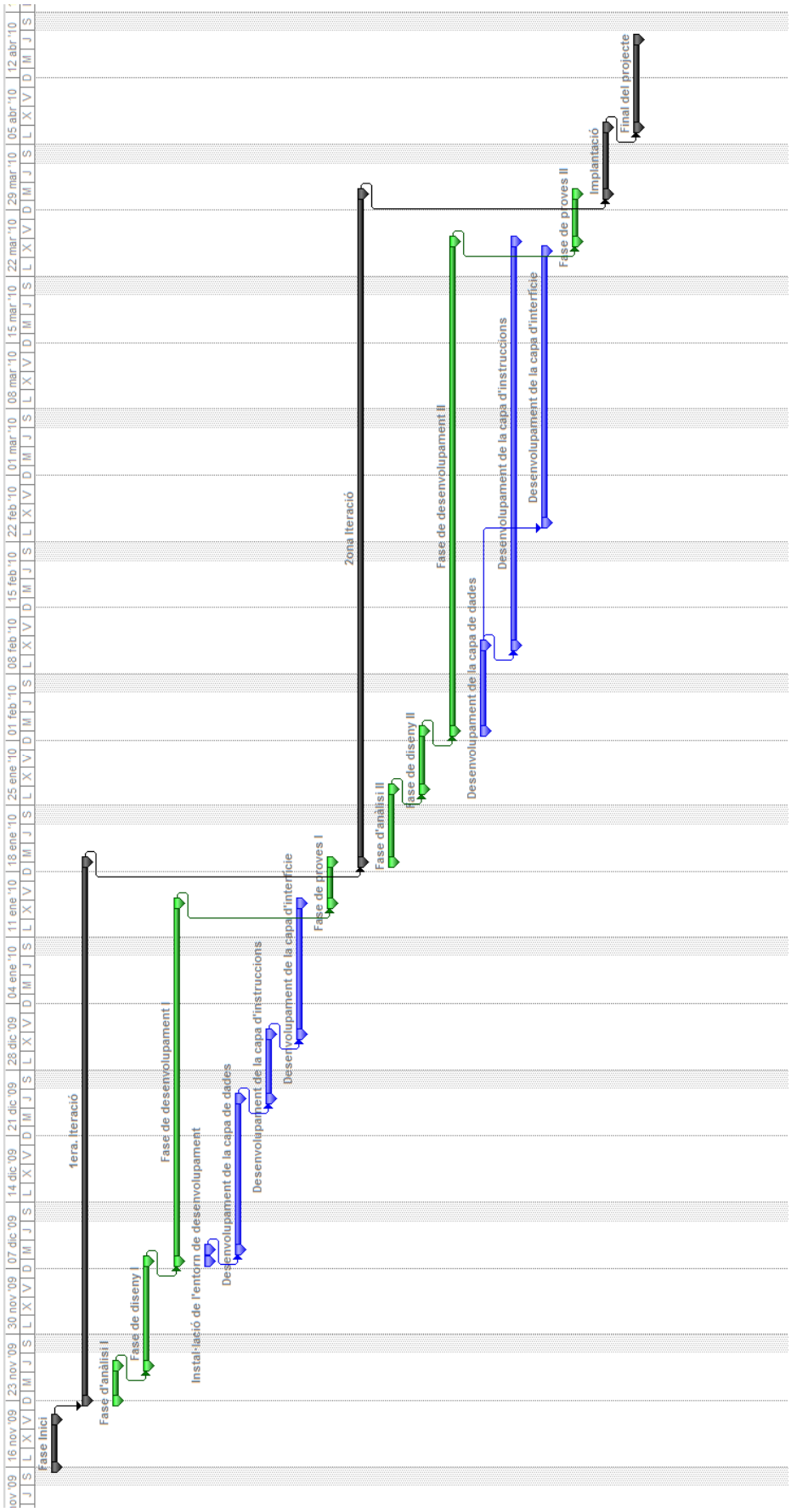
Un cop s'ha fet el disseny de l'aplicació s'ha tornat a fer la llista de les tasques s'han de dur a terme tenint en compte les decisions preses en el disseny. D'aquest manera es pot aproximar més aquesta planificació a el que finalment serà el temps dedicat a desenvolupar el projecte. Al detallar les tasques també s'explicita més clarament en quin punt es troba el projecte durant el seu desenvolupament.

Tasques del projecte

Nom de la Tasca	Durada	Inici	Fi
Fase Inici	**5d**	**16/11/2009**	**20/11/2009**
Inici del projecte: Assignació i matriculació del projecte	2h	16/11/2009	16/11/2009
Estudi de requisits des del punt de vista del usuari	2h	16/11/2009	16/11/2009
Estudi de viabilitat	20h	16/11/2009	20/11/2009
Aprovació d'estudi de viabilitat	1h	20/11/2009	20/11/2009
1era. Iteració	**41d**	**23/11/2009**	**18/01/2010**
Fase d'anàlisi I	**3,4d**	**23/11/2009**	**26/11/2009**
Anàlisi de requeriments funcionals	6h	23/11/2009	24/11/2009
Casos d'ús	4h	24/11/2009	24/11/2009
Estructura de dades	3h	25/11/2009	25/11/2009
Anàlisi de requeriments no funcionals	1h	25/11/2009	25/11/2009
Estudi de l'arquitectura de l'aplicació i marc tecnològic	3h	25/11/2009	26/11/2009
Fase de disseny I	**7,2d**	**26/11/2009**	**07/12/2009**
Estudi del llenguatge de desenvolupament	8h	26/11/2009	27/11/2009
Definició de perfils d'usuari	2h	30/11/2009	30/11/2009
Disseny de la interfície	4h	30/11/2009	01/12/2009
Disseny de la capa de instruccions	8h	01/12/2009	02/12/2009
Disseny de la capa de dades	2,8d	02/12/2009	07/12/2009
Disseny de la vaca	8h	02/12/2009	04/12/2009
Disseny de la granja	2h	04/12/2009	04/12/2009
Disseny de l'escenari de simulació	2h	04/12/2009	07/12/2009
Disseny entitat relació de la base de dades	2h	07/12/2009	07/12/2009
Fase de desenvolupament I	**27,6d**	**07/12/2009**	**14/01/2010**
Instal·lació de l'entorn de desenvolupament	1,4d	07/12/2009	08/12/2009
Instal·lació de màquina virtual de Java	1h	07/12/2009	07/12/2009
Instal·lació de servidor web, Apache Tomcat	2h	07/12/2009	08/12/2009
Instal·lació i creació de base de dades MySQL	4h	08/12/2009	08/12/2009
Desenvolupament de la capa de dades	12d	09/12/2009	24/12/2009
Desenvolupament de la classe vaca	50h	09/12/2009	22/12/2009
Desenvolupament de la classe granja	10h	23/12/2009	24/12/2009
Desenvolupament de la capa d'instruccions	4,8d	25/12/2009	31/12/2009
Desenvolupament de la simulació d'un dia de la granja	10h	25/12/2009	28/12/2009
Creació de granges	10h	29/12/2009	30/12/2009
Control d'accés d'usuaris	4h	31/12/2009	31/12/2009
Desenvolupament de la capa d'interfície	9,4d	31/12/2009	14/01/2010
Maquetat HTML de les interfícies	5h	31/12/2009	01/01/2010
Creació de la pàgina d'estils	5h	01/01/2010	04/01/2010
Interfícies d'accés a l'aplicació	2h	04/01/2010	05/01/2010
Interfícies d'estat de les vaques d'una granja	10h	05/01/2010	07/01/2010
Gràfiques d'evolució de la producció	20h	07/01/2010	13/01/2010
Càrrega dinàmica afegint funcionalitats Javascript	5h	13/01/2010	14/01/2010
Fase de proves I	**2,8d**	**14/01/2010**	**18/01/2010**

Nom de la Tasca	Durada	Inici	Fi
Proves de la capa de dades	2h	14/01/2010	14/01/2010
Proves de la capa d'interfície	5h	14/01/2010	15/01/2010
Proves funcionals	2h	15/01/2010	15/01/2010
Proves d'estrès	5h	18/01/2010	18/01/2010
2ona Iteració	**50,6d**	**19/01/2010**	**30/03/2010**
Fase d'anàlisi II	**5,2d**	**19/01/2010**	**26/01/2010**
Anàlisi de nous requeriments funcionals	12h	19/01/2010	21/01/2010
Casos d'ús dels nous requeriments	4h	21/01/2010	22/01/2010
Estructura de dades d'acord amb els nous requeriments	10h	22/01/2010	26/01/2010
Fase de disseny II	**4,8d**	**26/01/2010**	**01/02/2010**
Redefinició de perfils d'usuari	1h	26/01/2010	26/01/2010
Redisseny de la interfície	10h	26/01/2010	28/01/2010
Redisseny de la capa de instruccions	10h	28/01/2010	01/02/2010
Redisseny de la capa de dades	0,6d	01/02/2010	01/02/2010
Redisseny de la granja	1h	01/02/2010	01/02/2010
Redisseny de l'escenari de simulació	1h	01/02/2010	01/02/2010
Redisseny entitat relació de la base de dades	1h	01/02/2010	01/02/2010
Fase de desenvolupament II	**37,2d**	**02/02/2010**	**25/03/2010**
Desenvolupament de la capa de dades	7d	02/02/2010	10/02/2010
Desenvolupament del pool de connexions	5h	02/02/2010	02/02/2010
Desenvolupament de la classe vaca i d'accions de la vaca	20h	03/02/2010	08/02/2010
Desenvolupament de la classe granja i d'estat de la granja	5h	09/02/2010	09/02/2010
Desenvolupament de la classe d'entorn del curs	5h	10/02/2010	10/02/2010
Desenvolupament de la capa d'instruccions	30,2d	11/02/2010	25/03/2010
Creació de jerarquia de classes	5h	11/02/2010	11/02/2010
Creació de instruccions de control d'usuaris	2h	12/02/2010	12/02/2010
Desenvolupament de les funcionalitats usuari alumne	7,6d	12/02/2010	23/02/2010
Creació de accions sobre les vaques	10h	12/02/2010	16/02/2010
Revisió de la simulació	18h	16/02/2010	19/02/2010
Exportació de dades de la granja	10h	22/02/2010	23/02/2010
Creació de noves vaques	23h	12/02/2010	19/02/2010
Generació d'historial de successos de les vaques	10h	22/02/2010	23/02/2010
Desenvolupament de les funcionalitats del usuari administrador	3,6d	08/03/2010	11/03/2010
Creació de cursos	4h	08/03/2010	08/03/2010
Creació d'usuaris professor	4h	08/03/2010	09/03/2010
Copies de seguretat manuals de la base de dades	10h	09/03/2010	11/03/2010
Desenvolupament de les funcionalitats del usuari professor	10d	11/03/2010	25/03/2010
Gestió dels cursos	30h	11/03/2010	19/03/2010
Exportació de dades dels cursos	5h	24/03/2010	25/03/2010
Desenvolupament de la capa d'interfície	20,2d	24/02/2010	24/03/2010
Creació d'interfícies d'usuari	8d	24/02/2010	05/03/2010
Estat de la granja i gràfiques de producció	20h	24/02/2010	01/03/2010
Detalls de les vaques	10h	02/03/2010	03/03/2010
Taula d'inseminacions	10h	04/03/2010	05/03/2010
Creació d'interfícies d'administrador	3,2d	08/03/2010	11/03/2010
Creació d'interfícies de professor	3d	19/03/2010	24/03/2010
Detalls dels alumnes	15h	19/03/2010	24/03/2010
Fase de proves II	**3,4d**	**25/03/2010**	**30/03/2010**
Proves de la capa de dades	2h	25/03/2010	25/03/2010
Proves de la capa d'interfície	5h	25/03/2010	26/03/2010
Proves funcionals	5h	26/03/2010	29/03/2010
Proves d'estrès	5h	29/03/2010	30/03/2010
Implantació	**5d**	**30/03/2010**	**06/04/2010**
Instal·lació de l'aplicació	5h	30/03/2010	31/03/2010
Proves reals	5h	31/03/2010	01/04/2010
Documentació Instal·lació i manuals d'usuari	15h	01/04/2010	06/04/2010
Final del projecte	**7,4d**	**06/04/2010**	**15/04/2010**
Generació de documentació, memòria	30h	06/04/2010	14/04/2010
Tancament del projecte	2h	14/04/2010	14/04/2010
Defensa del projecte	5h	15/04/2010	15/04/2010
PROJECTE	**109d**	**16/11/2009**	**15/04/2010**

Diagrama de Gantt



Canvis en la planificació

Com a conseqüència del disseny de l'aplicació s'ha detallat el conjunt de tasques que s'han de dur a terme durant la implementació i s'ha fet una estimació més acurada en funció d'aquesta nova distribució de tasques.

Per aquest motiu el temps de realització del projecte s'ha ampliat i el nombre de tasques i subtasques ha crescut significativament. S'ha de destacar que a la 2a Iteració, què és quan es preveu la integració de la major part de les interfícies, aquestes s'implementaran paral·lelament a les funcionalitats que tenen associades a la capa d'instruccions.

5. IMPLEMENTACIÓ

5.1. INTRODUCCIÓ

A la implementació s'explica el resultat final de l'aplicació desenvolupada (Fig. 15). Es descriu la capa d'interfícies, d'instruccions i de dades. El disseny de cada capa és específic i per tant també s'han fet servir solucions específiques. Es descriu de quina manera es fan servir les eines que s'ha escollit per desenvolupar cada capa, i els cassos específics que per la seva complexitat mereixen una explicació més detallada.

Fig. 15 - Aspecte final de l'aplicació

5.2. INTERFÍCIES DELS PERFILS D'USUARI

Tal com s'especifica en el disseny les interfícies, aquestes s'implementen fent ús dels Servlets de Java. També s'aprofita el llenguatge Javascript i en concret la llibreria JQuery per afegir un cert comportament dinàmic a l'aplicació. I finalment l'aspecte va lligat a un arxiu d'estils CSS que especifica tot el que té a veure amb l'aspecte extern de l'aplicació.

JQuery

Per simplificar l'ús de les capacitats que té Javascript per afegir funcionalitat al codi HTML descarregat pel navegador del usuari, s'ha fet servir una llibreria

anomenada JQuery (JQuery Project, 2010). Per fer-la servir cal especificar la ruta des de on es carrega a la capçalera del document HTML. Just darrera d'aquesta especificació es pot carregar un altre fitxer Javascript que aprofiti les funcions definides en la llibreria JQuery. És pot incloure instruccions Javascript dins del propi codi HTML però segons els principis del disseny de pàgines web el que s'intenta és separar sempre l'estructura del document (HTML) de l'estil (CSS) i les funcionalitats agregades (Javascript) de manera que cada part es pugui editar amb independència (González, 2010). De fet és la mateixa arquitectura de capes feta servir per definir l'aplicació web però aplicada dins el nivell d'Interfícies.

La manera d'aconseguir-ho és etiquetant les estructures HTML perquè el CSS identifiqui la part de la web a la que afecta i el Javascript identifiqui les estructures a les que afegeix funcionalitat. JQuery simplifica aquets canvis en la funcionalitat proporcionant funcions per accedir i manipular elements HTML identificats per atributs propis del llenguatge HTML (Wilton-Jones, 2009).

En el següent exemple es mostren dos elements HTML definits per l'etiqueta SELECT. Aquest element representa un desplegable amb diferents opcions. Afegint funcionalitats en l'arxiu Javascript, aprofitant les funcions de les llibreries de JQuery, es pot recarregar la pàgina passant nous paràmetres cada cop que s'hi detecta un canvi en qualsevol dels dos objectes SELECT.

Codi HTML:

```
<HTML>
  <HEAD>
      <SCRIPT TYPE="text/javascript" SRC="jquery.js"></SCRIPT>
      <SCRIPT TYPE="text/javascript" SRC="/js/granja.js"></SCRIPT>
  </HEAD>
  <BODY>
      <SELECT NAME="filtro1">
            <OPTION VALUE="1">Opcion1</OPTION>
            <OPTION VALUE="2">Opcion2</OPTION>
      </SELECT>
      <SELECT NAME='filtro2'>
            <OPTION VALUE="1">Opcion1</OPTION>
            <OPTION VALUE="2">Opcion2</OPTION>
      </SELECT>
  </BODY>
</HTML>
```

Codi Javascript:

```
/*
 * Amb JQuery es pot afegir un comportament al document HTML un cop s'ha
 * acabat de descarregar tot el seu contingut al navegador del usuari.
 * Dins d'aquest comportament es poden definir tota mena d'accions que
 * s'executaran quan el document HTML estigui llest.
 */
$(document).ready(function () {
/*
 * Selecciona els elements SELECT que tinguin un atribut NAME que comenci
 * per "filtro". Els afegeix un comportament quan canvien amb la funció
 * change(), una funció que es defineix tot seguit com canviar la pàgina
 * que s'està mostrant per una a la que se li passen els nous valors
 * seleccionats en tots els SELECT com a variables.
 */
      $("SELECT[NAME^=filtro]").change(function() {
      /*
       * La forma d'accedir al valor dels SELECT és altre cop fent servir
       * funcions de JQuery que permeten agafar el valor d'aquets
      objectes
       * amb la funció val()
       */
            var novaRuta = "./ProfeGranjas";
            novaRuta += "?filtro1=" + $("SELECT[NAME=filtro1]").val();
            novaRuta += "&filtro2=" + $("SELECT[NAME=filtro2]").val();
            location.href = novaRuta;
      });
});
```

Enlloc de tornar a carregar la pàgina sencera també podem fer peticions a direccions d'Internet i carregar la resposta a parts de la estructura HTML. Per fer-ho JQuery proporciona la següent funció:

```
$.ajax({
      URL:"http://www.direccio.recurs",
      ASYNC : false,
      TYPE : 'POST',
      DATA : ({ var1:valor,var2:valor}),
      SUCCESS : function (resposta){
            $('#elementHtml').replaceWith(resposta);
      }
});
```

El paràmetre URL específica la direcció a la que es vol fer una petició. El paràmetre ASYNC estableix si es vol escriure la resposta conforme aquesta va arribant o un cop s'ha completat la resposta de la direcció especificada en el

paràmetre URL. El paràmetre TYPE estableix quin tipus de petició es fa a la direcció. Normalment les peticions son de tipus 'GET' però en el cas de formularis, si volem que les variables es passin sense que l'usuari les vegi es fa servir el tipus de petició 'POST'. El paràmetre DATA serveis per especificar les variables que es volen passar junt amb la petició i el valor que aquestes tenen. Finalment el paràmetre SUCCESS permet associar una funció al fet de que la petició s'hagi produït amb èxit i dins d'aquesta funció podem especificar que el contingut de la resposta es carregui en substitució d'algun element del codi HTML. El seu comportament es pot veure a la Fig. 16.

Fig. 16- Exemple de comportament AJAX, l'usuari pitja el botó vermell que fa una crida a la funció ajax de JQuery que carrega un quadre amb els detalls de la vaca seleccionada

D'aquesta manera s'aconsegueix una recarrega parcial de la interfície que té davant l'usuari i dóna la sensació d'estar davant d'una aplicació interactiva, no d'una pàgina web amb contingut estàtic.

Idiomes

Un cop dins de l'aplicació es té l'opció de mostrar el contingut en tres idiomes (català, castellà i anglès) per facilitar la comprensió a alumnes de tot arreu (Fig.

17). Tots els textos de l'aplicació es carreguen des d'un arxiu de text en funció del idioma seleccionat. Això facilita la modificació dels textos i simplifica la possibilitat d'afegir més idiomes.

Fig. 17 - Detall del selector d'idiomes

Aquesta implementació és possible gràcies la classe ResourceBundle de Java que en funció d'un paràmetre carrega a la memòria les variables d'un dels fitxers d'idioma de l'aplicació ([nom_del_fitxer]_[idioma].properties). Per tal de que el idioma no canviï es guarda la configuració a la sessió del usuari. És molt senzill mantenir les interfícies en diversos idiomes, només cal anar afegint les mateixes variables però amb diferents traduccions a cada un dels arxius. Per la mateixa raó també és molt senzill afegir idiomes. Aquesta opció és molt millor que mantenir els textos a la base de dades perquè es tracta d'un contingut estàtic subjecte a poques modificacions i el fet de guardar-se en fitxers fa que tècnicament sigui molt senzill fer canvis en els continguts.

Sessions d'usuari

Un altre dels factors importants de la interfície son les variables que es guarden en un objecte de la classe HttpSession. Aquesta classe permet guardar dades associades a un usuari de l'aplicació de manera que no cal anar passant paràmetres perquè aquesta recordi els valors de les variables. Aquestes sessions es guarden, en part, al propi Això és especialment útil per controlar les dades d'accés a l'aplicació (nom d'usuari, paraula de pas, perfil d'usuari) i per recordar en quin idioma està fent servir l'aplicació. Mentre l'usuari està a l'aplicació només cal comprovar que la sessió existeix i veure quins permisos d'accés te. En sortir de l'aplicació o després d'un temps d'inactivitat es destrueix la sessió.

Aquesta sessió està relacionada amb l'usuari gràcies al context del l'aplicació que només fa accessible les variables allotjades en aquesta sessió als Servlets

instanciats pel mateix usuari que les ha generat de manera que quan l'alumne tanca el navegador totes les dades guardades a la seva sessió es perden.

També es fa servir un mètode dels Servlets que permet guardar variables del context de l'aplicació per controlar si un mateix usuari es connecta des de dos ubicacions diferents al mateix temps i fa fora a una de les dos sessions.

Interfícies d'accés

Les interfícies d'accés son les úniques que no fan ús de la classe Servlet perquè el seu contingut és completament estàtic. Estan escrites en arxius HTML a l'arrel de directori on s'allotja l'aplicació.

La primera interfície és la que dóna accés a l'aplicació web (Fig. 18). Des d'aquest punt és possible accedir al formulari que permet als alumnes associats a un curs crear la seva granja.

La interfície per generar les granges té com a finalitat que l'alumne completi el registre del seu usuari establint la paraula de pas (Fig. 19). Aquest alumne a de conèixer el seu nom d'usuari i una paraula clau definida per identificar el curs al que l'alumne està associat.

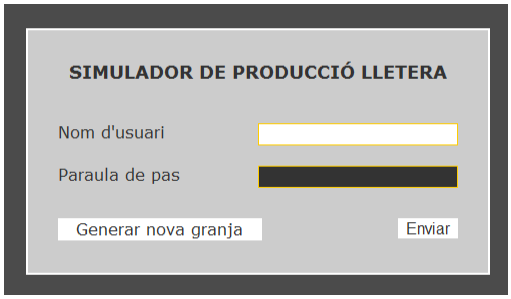


Fig. 18 - Formulari d'accés a l'aplicació

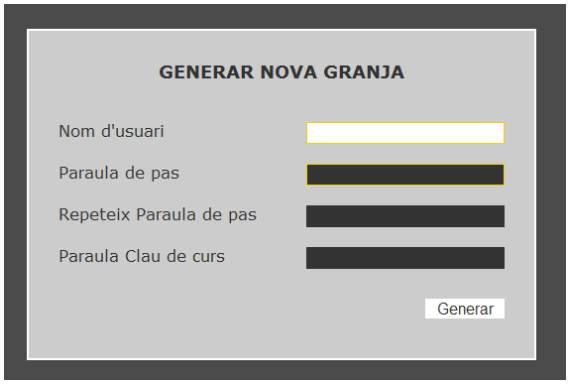


Fig. 19 - Formulari per generar granges

Totes dues interfícies afegeixen les dades del usuari a la sessió i redirigeixen a l'usuari a la interfície que correspongui al seu perfil.

Interfícies de l'usuari Alumne

El conjunt de pantalles a les que té accés l'alumne tenen per objectiu oferir el màxim d'informació possible sobre la granja a la que té accés i que aquesta informació sigui fàcilment accessible. També és important que les interaccions de l'usuari amb les vaques siguin àgils per facilitar l'aprenentatge de l'eina.

El menú lateral

De entre les opcions disponibles al menú lateral per l'usuari alumne (Fig. 20) destaquen les opcions:

- Exportar: permet a l'usuari descarregar un fitxer CSV per que tinguin la possibilitat de manipular les dades en exercicis proposats pel professor.
- Comprar: genera una nova vaca en el seu primer dia de lactància just després del seu primer part. Aquesta vaca no té ni història productiva ni de fertilitat de cap tipus. En principi una vaca d'aquestes característiques donarà una producció per sobre de la mitjana i per tal d'evitar que es millori la producció de la granja únicament d'aquesta manera es limita el número de vaques que es poden adquirir durant el curs.
- Simular: un cop estudiada la situació de la granja i fets el canvis sobre els animals, l'estudiant pot simular un dia de funcionament de la granja.
- Sortir: destrueix la sessió d'usuari i surt de l'aplicació web.

Fig. 20 - Menú lateral dels alumnes

Informe d'estat de la granja

Ja sigui a través del formulari d'accés a l'aplicació com després de generar la granja, al usuari Alumne se li mostra l'informe resum de la seva granja (Fig. 21). En aquesta pàgina l'estudiant veu l'estat inicial i l'estat actual del indicadors clau que determinen el bon funcionament de la granja.

Estat de la granja (5/1/2010)

Dada	Inicial	Actual
Producció total de Llet (l.)	3143.94	3165.67
Producció mitjana (l./vaca)	31.43	31.65
Inseminacions	649	649
Inseminacions exitoses	199	199
Index de Fertilitat	0.3	0.3
Vaques adquirides	0	2
Vaques sacrificades	0	2
Número total de vaques	100	100
Promig durada de secat	60.0	60.0
Dies simulats	0	4

Fig. 21 – Informe d'estat de la granja

També disposa d'unes gràfiques a la part inferior (Fig. 22) que informen de la tendència productiva mitjana de les vaques en funció de si estan en la seva primera lactància o no. Aquesta gràfica es construeix superposant els valors de producció de totes les vaques per als mateixos dies de lactància, de manera que el núvol de punts resultant és una aproximació a una corba productiva mitjana.

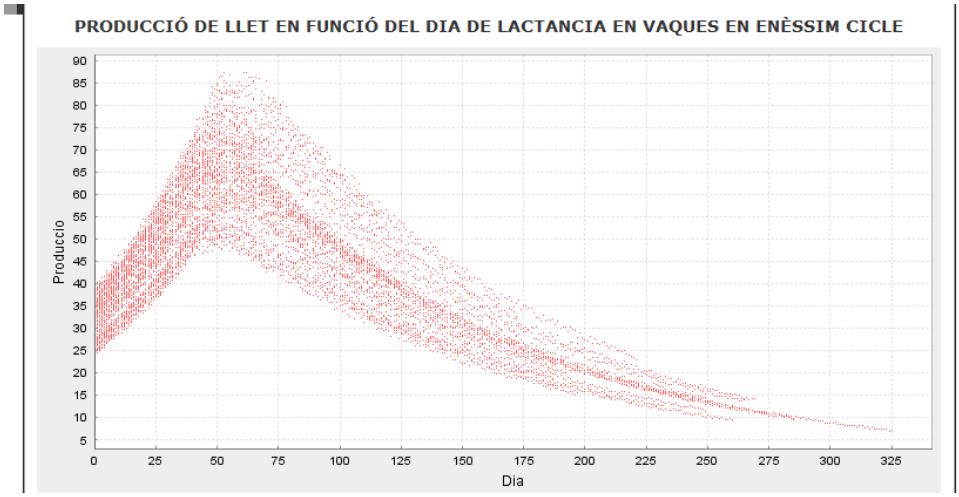


Fig. 22 – Detall d'una de les gràfiques en el informe d'estat de la granja

Aquest informe d'estat és sempre el primer impacte que rep l'usuari que explora la granja i des d'aquí ja es pot moure per les opcions de navegació que ofereix el menú lateral.

Llistat de les vaques de la granja

Aquest llistat (Fig. 23) dóna accés a tota la informació relacionada amb les vaques de la granja. Es pot consultar l'estat de les vaques, comprovar-ne la producció de llet o els nivells d'una sèrie d'indicadors clau del seu estat.

Fig. 23 - Llistat de les vaques

Aquesta interfície dóna a l'alumne la possibilitat d'interactuar amb les vaques a través del botó detalls que hi ha a cada fila del llistat. Un cop pitjat el botó es desplega una vista gràfica de l'evolució en la producció de la vaca i les accions disponibles per l'exemplar examinat (Fig. 24).

Fig. 24 - Detall de la gràfica i les accions disponibles per a una vaca

També es té accés a un l'històric de successos rellevants relacionats amb l'animal (Fig. 25). En cas de que la vaca estigui en període de zel, just a sota de les accions disponibles (Fig. 24) es mostra una petita ajuda visual que simula l'observació d'un comportament específic en una vaca real. Aquesta ajuda visual

és una pista que indica als alumnes el millor moment per dur a terme la inseminació.

Fig. 25 - Detall de l'històric de successos d'una vaca

Hi ha accions que no s'executen de forma immediata com son assecar o inseminar la vaca. Si l'alumne vol, per exemple, inseminar la vaca, és destaca la fila del llistat on es mostren les dades de la vaca perquè l'usuari sigui conscient de que hi ha accions pendents d'execució per aquella vaca. Després quan es simuli un dia a la granja, tots les accions pendents s'executen i deixen d'estar destacades les vaques amb accions pendents. Aquestes accions es guarden a la base de dades per tant és possible planificar una sèrie d'accions sobre les vaques i deixar la execució de les accions durant una simulació per més endavant.

Taula de Fertilitat

A través del menú lateral es pot accedir a una taula amb totes les inseminacions dutes a terme a la granja. Ofereix informació de quan es van dur a terme i amb quin exemplar. Al mateix temps indica a l'usuari quines d'aquestes inseminacions han tingut èxit destacant-les amb color vermell (Fig. 26).

Vaques	Intent: 1	Intent: 2	Intent: 3	Intent: 4	Intent: 5	Intent: 6	Intent: 7	Intent: 8
Vaca: 16397	17/1/2009	7/2/2009						
Vaca: 16398	10/1/2009	17/12/2009						
Vaca: 16399	19/1/2009	11/2/2009	3/3/2009	23/3/2009	13/4/2009			
Vaca: 16400	14/1/2009	20/12/2009						
Vaca: 16401	25/1/2009	22/12/2009						
Vaca: 16402	21/1/2009	14/2/2009	5/3/2009					
Vaca: 16403	17/1/2009	6/2/2009	28/2/2009	23/3/2009	14/4/2009	4/5/2009	28/5/2009	19/6/2009

Fig. 26 - Taula de fertilitat

Aquesta informació a d'ajudar tant a l'alumne com al professor a fer un seguiment de la política d'inseminacions del primer i comprovar si està desaprofitant ocasions per mantenir un bon índex de fertilitat a la granja.

Interfícies de l'usuari Professor

El conjunt de pantalles a les que té accés el professor tenen per objectiu oferir el màxim d'informació possible sobre els cursos als que té accés i que aquesta informació sigui fàcilment accessible. L'avaluació de l'alumne és important i per tant la navegació i les eines han de facilitar-ho.

El menú lateral

El menú lateral de la interfície del professor (Fig. 27) té una doble funció: per una banda és el selector del curs sobre el qual es vol treballar i per l'altra dóna accés a tot la informació del curs seleccionat. També disposa d'una opció per exportar les dades de les granges d'un curs en un fitxer CSV.

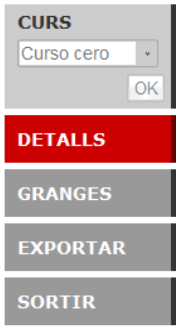


Fig. 27 - Menú lateral de la interfície de professor

Detalls i variables del grup

Aquest formulari (Fig. 28) permet fer canvis en el nom, la paraula de pas o l'estat del curs. Després més de cinquanta variables permeten el control total de l'entorn de simulació de les granges d'un curs. Un control que es pot exercir abans de que els alumnes generin les granges: per exemple definint el nombre d'animals que tindrà cada estudiant. O durant la simulació: per exemple limitant el nombre de simulacions per sincronitzar el ritme de l'aula o canviant un multiplicador de producció per influir en els resultats productius.

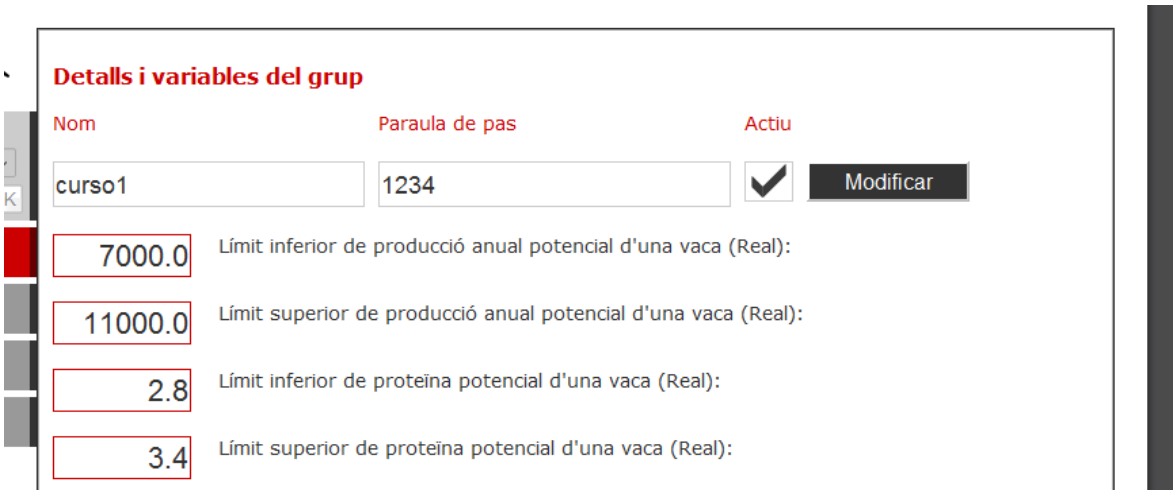


Fig. 28 - Configuració del curs

Granges del curs

En aquest apartat de l'aplicació el professor té una visió general de les granges dels alumnes. Disposa d'una sèrie de desplegables per mostrar i ordenar les granges segons un seguit d'indicadors clau de progrés dels alumnes (Fig. 29). A la part inferior de l'informe disposa de formularis per afegir alumnes ja sigui d'un en un o carregant un arxiu CSV amb les dades dels estudiants (Fig. 30).

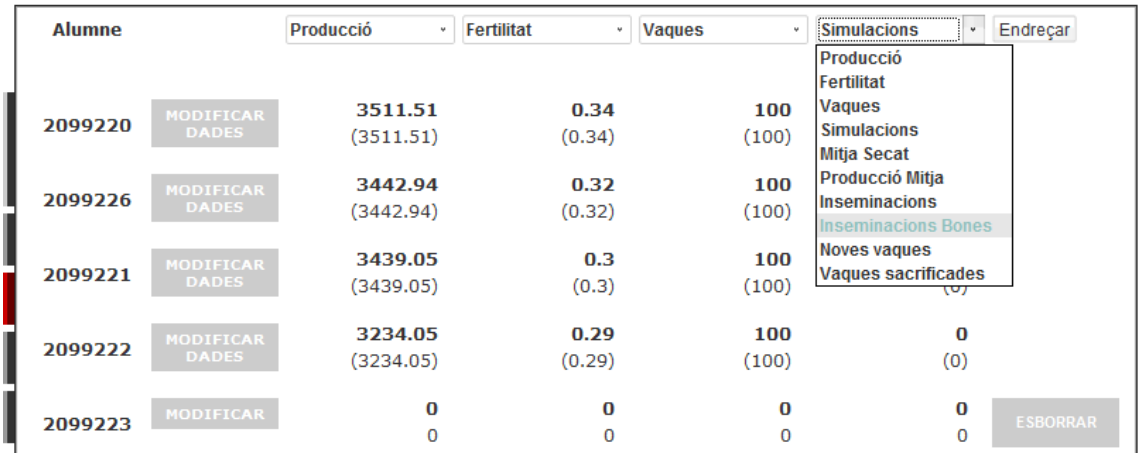


Fig. 29 - Granges del curs

Al peu de la interfície es mostra una barra de progrés que indica el percentatge dels participants en el curs que ha generat la seva granja respecte del total d'estudiants apuntats (Fig. 30).

Fig. 30 - Detall del peu de l'informe de les granges del curs

Aquesta pàgina és també el punt d'accés per:

- visitar en detall qualsevol de les granges del curs com si es tractés del propi alumne però sense les opcions per interactuar amb la granja,
- modificar les dades d'accés dels estudiants,
- esborrar els usuaris.

Totes aquestes opcions es carreguen dinàmicament a través dels botons disponibles per cada granja (Fig. 31).

Fig. 31 - Detall de les opcions de modificació de les dades dels alumnes

Interfícies de l'usuari Administrador

El conjunt de pantalles a les que té accés l'Administrador inclouen les funcionalitats disponibles per als professors. A més disposa d'un paquet d'opcions extres destinades a la gestió dels cursos, els usuaris professors i les variables d'entorn de simulació per defecte per a qualsevol nou curs que es vulgui crear en el futur. També disposa de l'opció de fer una copia de seguretat de la base de dades de l'aplicació.

El menú lateral

El menú lateral (Fig. 32) de la interfície d'administrador és similar a la de professor però inclou més opcions per afegir les funcionalitats extres de que disposa el responsable de l'aplicació. A més en el desplegable de cursos hi apareixen tots. D'aquesta manera l'administrador pot navegar per totes les interfícies i té accés a totes les dades emmagatzemades a l'aplicació.

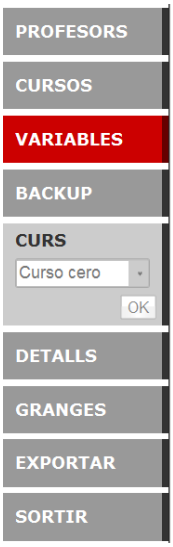


Fig. 32 - Menú lateral de la interfície d'Administrador de l'aplicació

Gestió d'usuaris de perfil professor

Es tracta d'una llista dels professors amb opcions per modificar-ne les dades, afegir-ne de nous o esbarrar-los (Fig. 33).

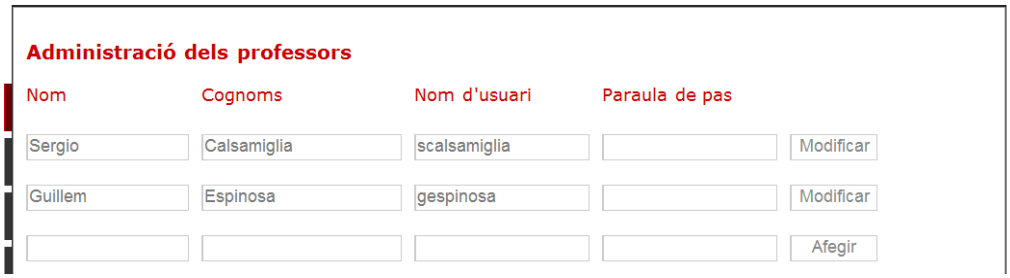


Fig. 33 - Gestió d'usuaris

Gestió de cursos

És una interfície que mostra la informació bàsica dels cursos així com els professors associats. Permet modificar-ne les dades i vincular o desvincular

professors als cursos (Fig. 34). La casella de verificació verda deixa oberta la possibilitat de generar les granges per part dels alumnes. D'aquesta manera es pot decidir quan està permès crear les granges (normalment els primers dies del curs) i quan no.

Fig. 34 - Gestió de cursos

Variables per defecte del sistema

És un formulari molt similar al que poden fer servir els professors per ajustar les variables d'entorn de simulació d'un curs. En aquest cas, però, les variables que es poden ajustar son els valors per defecte quan es genera un nou curs. Això permet especificar una configuració inicial concreta per a tots els cursos (Fig. 35). Aquesta funcionalitat facilita que en un principi el professor que ha definit les especificacions de la simulació pugui afinar les variables inicials de cara a que a principi de curs tots els professors tinguin unes configuracions inicials comunes per tots els grups.

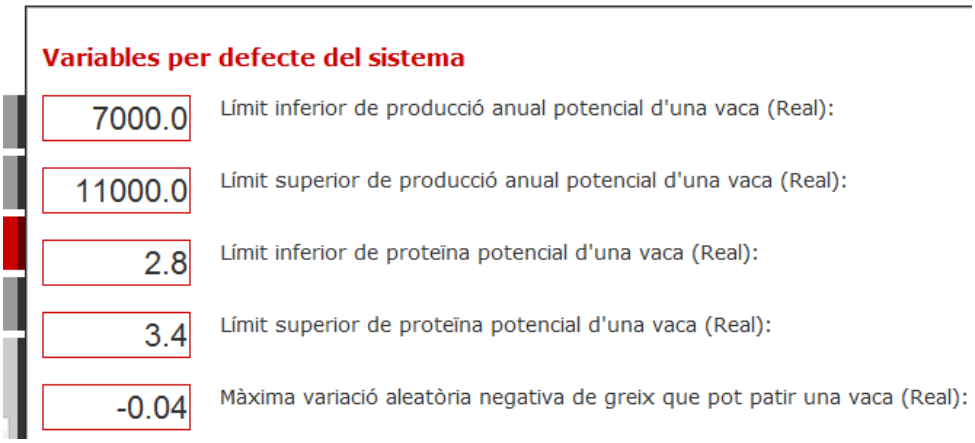


Fig. 35 - Variables per defecte del sistema

5.3. LA CAPA D'INSTRUCCIONS

A l'hora de dissenyar l'estructura ja es va definir un capa de l'arquitectura que implementin instruccions. Ja que l'aplicació funciona com una pàgina web. Es basa doncs en petits intercanvis d'informació gestionats per multitud de petites aplicacions web del servidor que son els Servlets. Aquestes instruccions van des de casos tan senzills com modificar les dades d'un usuari com a d'altres més complexes com és el generar una granja amb un escenari de simulació. D'altres instruccions es simplifiquen molt gràcies al disseny dels objectes que son els que finalment fan la feina de simular.

Apache Tomcat i els Servlets

Aquets Servlets estan disponibles com a serveis web i l'estructura que permet això és el servidor web Apache Tomcat. Gràcies a un arxiu anomenat web.xml es pot associar les classes amb serveis de la web accessibles a partir de l'arrel d'aquest servidor web. La manera de associar els Servlets amb els serveis és la següent (Apache, 2010):

```
<servlet>
      <servlet-name>Nom_Classe_Servlet</servlet-name>
      <servlet-class>Classe_Servlet</servlet-class>
</servlet>
<servlet-mapping>
      <servlet-name>Nom_Classe_Servlet</servlet-name>
      <url-pattern>/Nom_Servei_Web</url-pattern>
</servlet-mapping>
```

Un cop associat el Servlet a un servei Web cal sobreescriure el mètode doGet() o doPost() que hereta de la classe i que depèn del tipus de petició web a la que es vulgui donar resposta amb aquest servei. Les peticions POST son més segures però les peticions GET son més flexibles.

Un cop s'arrenca el servidor aquest inicia com a serveis tots el Servlets que estiguin definits en l'arxiu web.xml (Fig. 6), es carreguen en memòria les classes corresponents i a partir d'aquest moment per casa petició que rep algun d'aquets serveis és crea un fil d'execució del mètode doPost() o doGet().

Aprofitant l'estructura simple i alhora molt potent d'aquestes classes de Java, s'han implementat tot un seguit de petits programes que atenen a peticions molt concretes i en retornen resultats o bé simplement processen la informació que reben. Treballar amb un nombre moderat de Servlets carregats en memòria del servidor té les seves recompenses. L'accés dels usuaris és molt ràpid i conjuntament amb l'aprofitament de la tecnologia AJAX de carrega dinàmica de continguts de forma asíncrona millora la facilitat d'ús de l'aplicació. La seva implementació no és molt complexa i en alguns casos hereten utilitats de classes que agrupen mètodes comuns com és la connexió a la base de dades o la gestió de les sessions d'usuari.

La classe Historial

Un dels casos més complexes i que més feina han donat és la instrucció que genera una nova granja i un seguit de vaques amb un història simulada automàticament de forma aleatòria. El problema a resoldre no és crear un nombre determinat d'objectes sinó més aviat generar tot el volum de dades necessari per que cada alumne treballi amb un escenari diferent del dels seus companys. Aquest procés de generar dades és molt costós en recursos del sistema i cal una estructura que ho agilitzi per evitar que el primer dia de simulació els alumnes estiguin esperant durant un temps excessiu que es generin les seves granges.

La solució passa per fer ús de la classe Runnable. En el conjunt de classes del diagrama es veu una classe funcional Historial (Fig. 38). Aquesta classe implementa la classe de Java Runnable que permet que la crida al mètode run() d'aquesta classe sigui un fil d'execució paral·lel a l'execució principal (Fig. 36). Aquesta funcionalitat s'aprofita en el moment de generar un nombre molt important de simulacions automàtiques que fent ús de fils d'execució paral·lels agilitza notablement el procés (Franco, 2000).

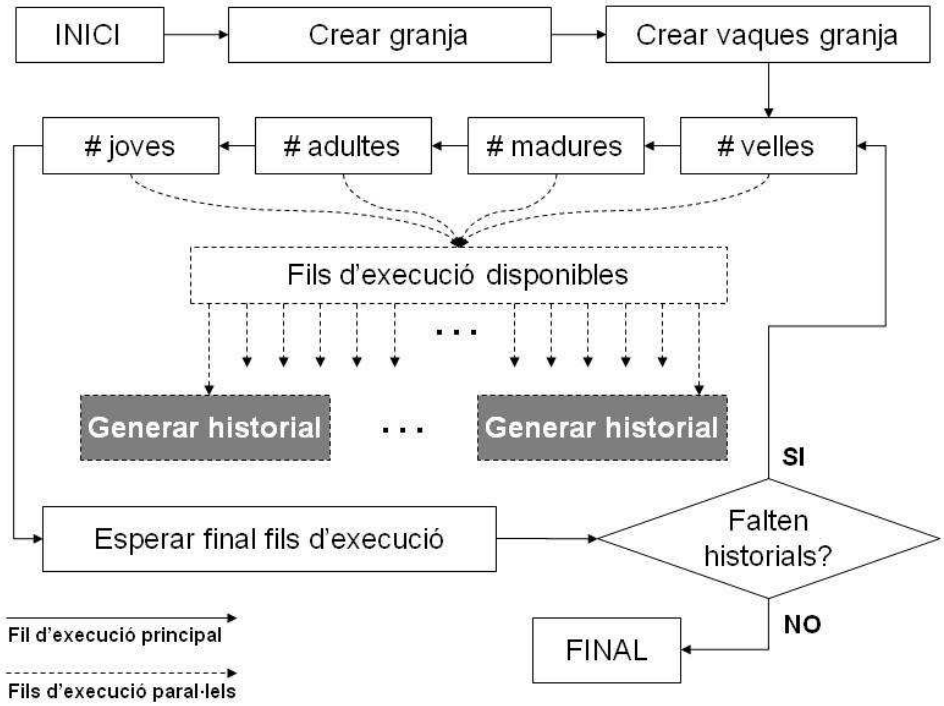


Fig. 36 – El paper de la classe Historial quan es genera la granja

Amb l'adopció d'aquesta implementació la creació d'una granja triga un temps aproximat d'un minut. Com a conseqüència negativa, cal controlar la carrega del sistema per no acumular connexions de la base de dades, espai de memòria i temps de processador per tal d'aconseguir un ús dels recursos balancejat. Aquesta carrega dels recursos es pot afinar mitjançant unes constants definides tant a la classe Historial com a la classe Granja.

Gràfiques i JFreeChart

La funcionalitat de dibuixar gràfiques es complica pel fet d'haver decidit treballar amb un paquet de classes extern de la que no se sap la seva implementació. Es tracta de JFreeChart, un paquet que ofereix un amplíssim ventall d'eines per generar gràfiques de tot tipus. L'esforç es veu recompensat amb unes gràfiques molt vistoses que resolen molt bé el requeriment funcional (Object Refinery Limited, 2009).

A partir d'un objecte d'aquest paquet JFreeChart es pot carregar les dades que es volen mostrar a la gràfica per a unes coordenades específiques. Després aquest objecte es fa servir per construir un objecte Chart (per exemple una

gràfica de barres Fig. 37) del tipus desitjat. Finalment mitjançant un objecte Plot que dibuixa la gràfica es pot decidir l'aspecte de la gràfica (ChuWiki, 2009).

Un conjunt d'utilitats del paquet ens permet decidir el format de sortida de les dades. Aprofitant el comportament dels Servlets s'utilitza un objecte del paquet JFreeChart que escriu una imatge com a resposta de la petició web (Hablutzel, 2009). Aquesta imatge es carrega fent una petició al Servlet que la dibuixa des de la interfície que mostra la imatge a l'usuari (per exemple el detall de l'evolució en la producció de llet d'una vaca). En el fons el que fa l'aplicació és encadenar peticions a diferents Servlets per mostrar una única interfície a l'usuari.

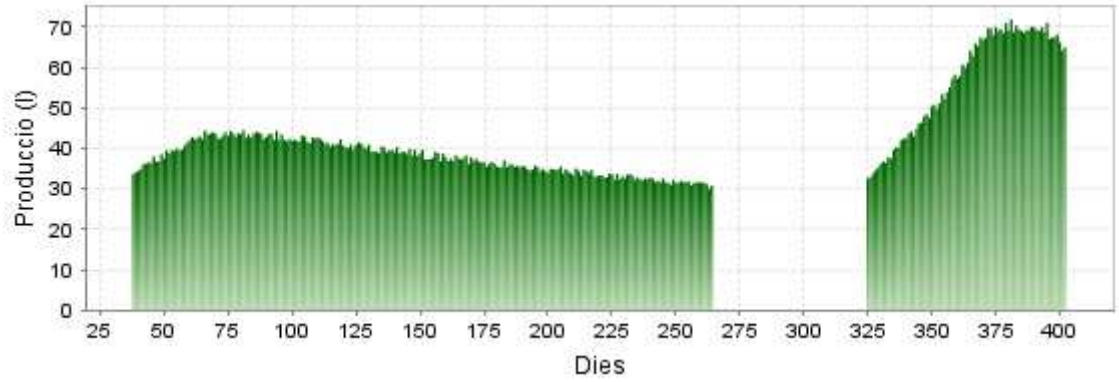


Fig. 37 - Exemple de gràfica de producció de llet generada a través de JFreeChart

5.4. LA CAPA DE DADES

L'estructura funcional que s'ha implementat té una sèrie de classes que funcionalment son objectes: Vaca, Acciones, Granja, EstadoGranja i EntornoCurso. Aquestes classes es relacionen com es mostra al diagrama (Fig. 38) i tenen una correspondència amb les taules i atributs de la base de dades (Fig. 39).

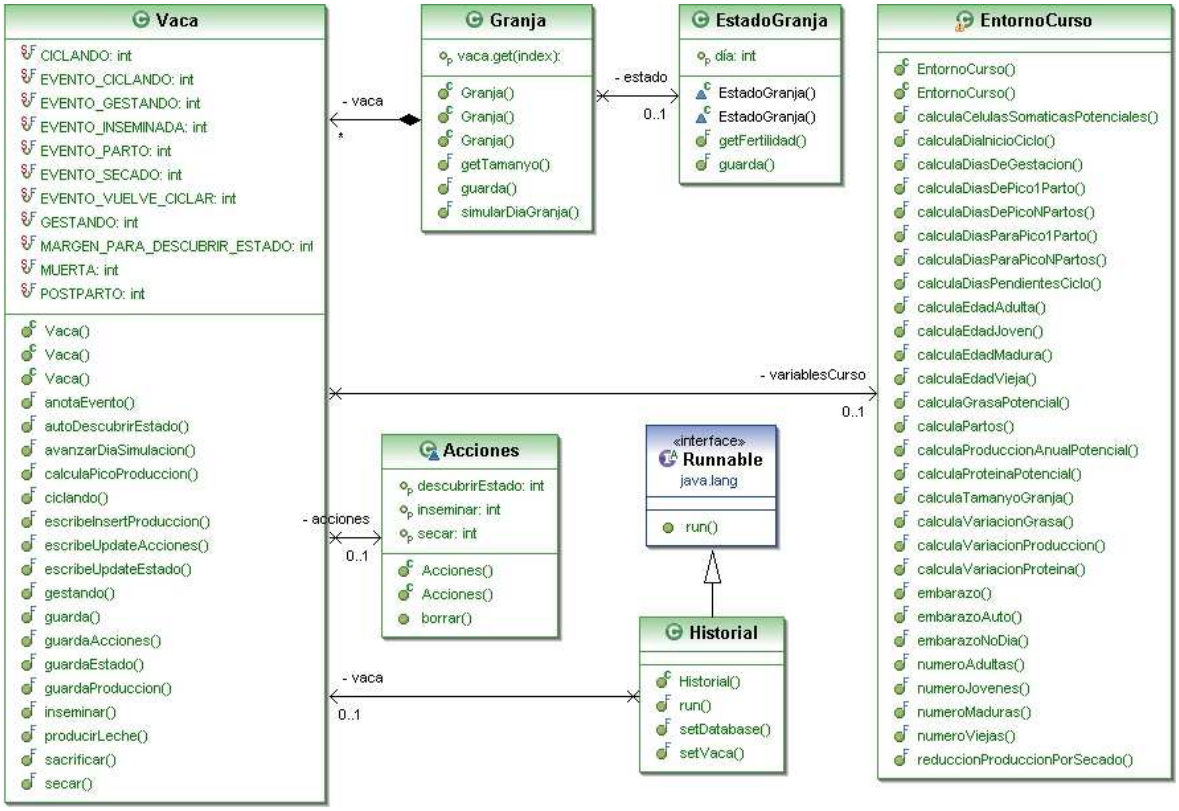


Fig. 38 – Diagrama de classes de la capa de dades i la classe Historial

S'ha de destacar la classe EntornoCurso que conté totes les variables que concreten l'entorn de simulació i que per l'experiència adquirida durant el projecte és probable que de cara a futures ampliacions hagi de créixer més. Dins d'aquesta classe es defineixen les cotes dels intervals de possibles valors que poden tenir les variables que defineixen les vaques o les granges. També hi ha mètodes que retornen valors aleatoris dins d'aquets intervals.

Una altra classe a destacar es la Vaca que conté, a part dels atributs que defineixen el seu estat, bona part dels mètodes que implementen la simulació d'un dia de la vaca. En principi la lògica suggeria associar tots el mètodes que provoquen canvis d'estat en la vaca a la pròpia classe vaca. Però al llarg del projecte s'han vist conceptes com els patrons de disseny que suggereixen que de cara a futures ampliacions de l'aplicació hagués estat més útil separar la implementació del objecte vaca de totes les funcionalitats associades a ella. A més la complexitat de simular un dia de la vaca fa que dins de la classe vaca hagin proliferat tota mena de mètodes auxiliars per estalviar codi i facilitar la comprensió del seu funcionament intern.

La base de dades

La implementació de la base de dades es basa en el disseny dut a terme al principi del projecte (Fig. 14) però s'han afegit taules necessàries per respondre a requeriments que s'han inclòs a mida que el projecte anava avançant. L'estructura final es pot veure en el següent diagrama (Fig. 39).

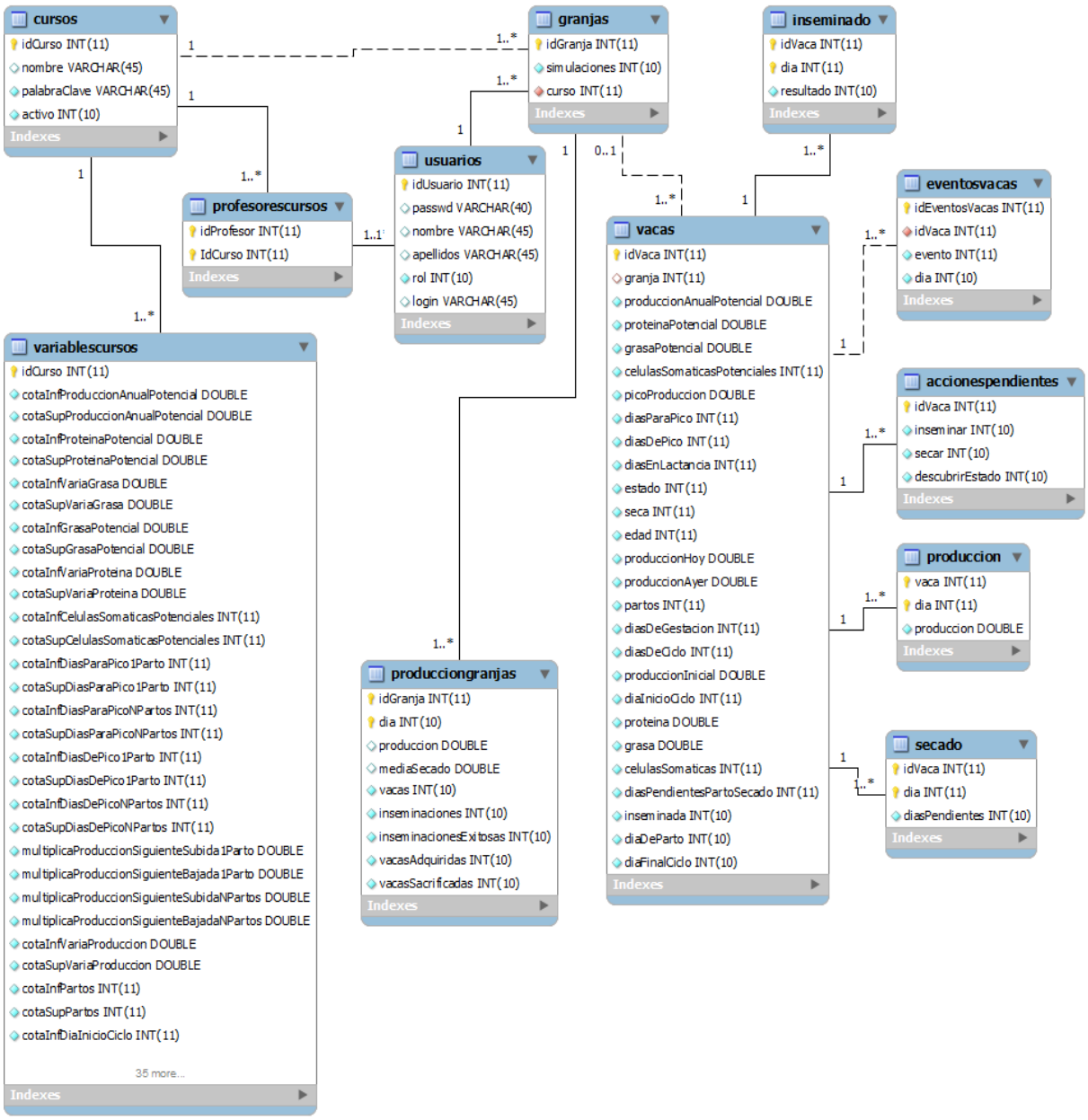


Fig. 39 - Model UML de la Base de Dades

En el diagrama, les claus grogues a l'esquerra dels atributs representen les claus primàries de les taules. Els rombes de color vermell representen les claus foranies. Les línies contínues relacionen taules amb referències a altres taules

dins les seves claus primàries. Les línies discontínues representen les referències a d'altres taules per part d'atributs que no formen part de la clau primària (Oracle, 2010).

El pool de connexions i el paquet DB

Una part molt interessant de la implementació ha sigut desenvolupar un sistema de gestió de connexions amb la base de dades. Normalment es perd molt de temps demanant una connexió al servidor, obrint-la i després tancant-la per això existeixen el que s'anomena pools de recursos. En aquest cas el recurs és una connexió a la base de dades i el que és fa és reservar un número determinat de connexions a la base de dades obertes i les va assignant a les peticions al servidor. D'aquest manera els temps de resposta de les peticions son més petits i es protegeix a la base de dades d'un ús excessiu de connexions (Casas, 2008).

Aquest pool de connexions es pot definir en el context del programa que atén les peticions dels usuaris. El servidor de peticions per aplicacions web amb Servlets de Java és l' Apache Tomcat. Aquest servidor disposa d'una eina per administrar recursos limitats definint una reserva del tipus de recurs que es vulgui. En el cas del Simulador s'ha definint un recurs que és el pool de connexions. En aquesta definició s'estableix el nombre de connexions reservades, el nombre de connexions en espera i el temps màxim que s'ha d'esperar una petició per que se li assigni una connexió.

Ja només cal definir una classe que s'ha anomenat DbConnection dins d'un paquet al que s'ha anomenat DB. Aquesta classe conté els mètodes que proporcionen accés a la base de dades i funcionalitats associades a aquestes connexions com per exemple fer una consulta a la base de dades o alliberar una connexió.

La classe Utils

De entre totes les classes que s'han implementat n'hi ha una que destaca en relació a la base de dades. La classe Utils conté mètodes més o menys utilitzats per altres classes però sense una relació directa amb aquestes. També conté

valors constants per a l'aplicació als quals es fa referència des de les altres classes.

Principalment té dos tipus de mètodes. El primers tenen per objectiu afegir funcionalitats relacionades amb els Sistema que allotja l'aplicació: un mètode que detecta el sistema operatiu de la maquina on s'allotja i un mètode que detecta el consum de memòria de l'aplicació que s'ha fet servir per fer proves. Els segons estan relacionats amb la conversió de textos de les interfícies a la base de dades o a l'inrevés. Serveixen per escapar caràcters susceptibles de no mostrar-se correctament a les interfícies o de provocar errors en l'aplicació. Al mateix temps serveixen per acotar la mida dels paràmetres d'entrada de l'aplicació perquè no superin els límits dels atributs especificats en la base de dades.

És per aquest segon conjunt de funcionalitats que aquesta classe té una estreta relació amb la base de dades. Un dels atacs més habituals a les aplicacions web és posar caràcters d'escapament en els formularis de les interfícies per executar sentencies SQL dins de la base de dades. Amb aquestes funcions s'impedeix que això succeeixi.

6. PROVES

De la mateixa manera que s'ha estructurat tot el projecte en funció de l'arquitectura de l'aplicació, també s'han fet proves destinades a comprovar cada nivell de l'arquitectura. El fet de fer servir diferents combinacions de tecnologies aconsellava també comprovar el seu correcte funcionament tant per separat com funcionant globalment.

6.1. PROVES D'INTERFÍCIES

Per provar les interfícies s'ha tingut en compte que actualment no hi ha un programari que s'imposi per damunt dels demès i sigui la eina de referència per la navegació de pàgines web. Això ha provocat que s'haguessin de fer proves contínuament per comprovar de quina manera es visualitzen les interfícies en funció de l'aplicació que pugui fer servir l'usuari per visualitzar-les.

S'han provat tant els estils com l'estructura HTML com les funcionalitats Javascript amb el següent programari:

Mozilla Firefox 3.6	Internet Explorer 8.0	Internet Explorer 7.0	Internet Explorer 6.0	Opera 10.5
Compatible	Compatible	Parcialment compatible	Incompatible	Compatible

A part de comprovar la forma en que es comporten les interfícies en cada navegador també s'ha comprovat les comunicacions entre el Javascript i els Servlets i s'ha repassat el comportament dels arxius CSS fent servir les eines de detecció de problemes dels diferents navegadors: "Consola de Errores" de Firefox, "Herramientas de Desarrollo" de Internet Explorer i "Dragonfly" de Opera.

S'ha tingut també cura de que el codi HTML generat tingués una correcta notació per detectar si hi havia errors en la seva estructura. Els errors en el llenguatge HTML no impedeixen que la pàgina web que descriuen no s'executi però si poden provocar desordres en la forma en que es mostra la informació. Per facilitar la detecció d'errors es molt important fer servir una notació molt clara.

6.2. PROVES A LA CAPA D'INSTRUCCIONS

A aquesta capa s'ha provat que es complissin les funcionalitats i les seves restriccions. S'han fet proves per cada instrucció un cop s'ha implementat. Això es possible perquè es tracta de classes amb funcionalitats molt concretes que son fàcils de provar.

A partir d'aquestes provés s'han detectat errors en la longitud de les cadenes de text que es guardaven a la base de dades i amb alguns caràcters especials com l'apòstrof o la "\". Com a conseqüència d'aquestes proves s'han afegit els mètodes que escapen les cadenes de caràcters a la Classe Utils. Després es va detectar que era necessari fer el procés invers si s'exportaven les dades a un fitxer de text.

Hi ha un mètode que ha estat sotmès a més proves que cap altre. Es tracta del mètode que genera les granges. A aquest mètode se l'ha sotmès a tota mena de provés per determinar la carrega que representava per el sistema el generar els historials d'una granja. S'ha comprovat el temps que trigava i la memòria consumida durant les proves. S'ha provat a fer granges d'una sola vaca fins a una granja de 20.000 animals. Com a conseqüència de les proves es va modificar el constructor de noves granges a la classe Granja per equilibrar la carrega del sistema i després de detectar una fuga de memòria que provocava el col·lapse d'una maquina amb 4 GB de memòria RAM quan es generava granges amb una mida de més de 800 vaques. També s'han fet canvis per compartir una única connexió a la base de dades durant tot el procés de creació de la granja. I la idea de fer servir el pool de connexions s'ha originat quan es va constatar que es requerien masses connexions a la base de dades en el procés de creació d'una nova granja.

S'ha de tenir en compte que durant la creació de la granja es fan més simulacions de les que farà cap estudiant durant un curs. Aquest fet converteix aquest mètode en un excel·lent banc de proves de l'aplicació i per la mateixa raó una font d'errors cada cop que s'han modificat classes relacionades amb la simulació.

6.3. PROVES A LA CAPA DE DADES

A nivell de la capa d'objectes hagués estat molt interessant implementar les proves d'unitat però finalment no hi ha hagut temps i la pròpia complexitat a nivell de mètodes feia que aquest fos un objectiu molt costós pel que fa al temps.

Tot i així les primeres proves de funcionament es van fer amb les classes d'aquesta capa. Al tractar-se de classes de Java que es poden instanciar amb qualsevol codi de prova (no cal instal·lar un Servidor Web per comprovar funcionalitats) s'ha anat fent proves amb aquestes classes pràcticament des del primer dia d'implementació. S'ha provat cada mètode primer amb valors fixes per les variables que defineixen les vaques i més endavant amb les variables generades per la classe de les variables d'entorn de l'aplicació. S'han fet proves en els canvis d'estat i s'han detectat problemes amb la periodicitat de les inseminacions que s'han corregit.

També s'ha provat l'evolució de la producció on s'han detectat errors en la seqüència en la que es criden els mètodes corresponents a cada estat de la vaca i s'han corregit. S'han mostrat les gràfiques d'evolució de la producció al professor de veterinària per que determinés si representaven el resultat esperat. Després de veure-les el professor va suggerir afegir una petita variació aleatòria a la producció diària per donar més realisme a les gràfiques.

S'han provat els constructors que creen objectes a partir de la base de dades i s'ha creat mètodes per generar les sentencies de inserció i actualització dels objectes a la base de dades per evitar errors comuns.

S'ha preparat cada consulta a la base de dades en una consola SQL. Quan s'ha aconseguit el resultat esperat s'ha substituït els camps per variables i s'han afegit missatges que imprimissin les consultes SQL per provar cassos concrets directament a la consola SQL. S'ha posat a prova la integritat referencial de les taules i s'ha migrat en varies ocasions la base de dades per comprovar que la copia de seguretat de les dades fos correcte.

S'ha fet seguiment del consum de recursos a la base de dades per comprovar un ús excessius de transaccions, de la memòria per a consultes i del consum de connexions. A nivell de connexions a la base de dades s'ha corregit un error que te el seu origen en declarar la connexió com a variable global dels Servlets enlloc de al fil d'execució.

6.4. PROVES FUNCIONALS

S'ha provat l'experiència de simular 365 dies (el nombre de simulacions que en principi el professors volen fer simular als estudiants). Durant aquest 365 dies s'ha fet el possible per millorar a màxims la producció per imitar el comportament d'un alumne. S'ha constatat que és molt difícil millorar la producció i a partir d'aquesta experiència el professor ha determinat afegir la possibilitat de sacrificar vaques i comprar-ne de noves com a part de l'exercici de simulació.

6.5. PROVES REALS

Un cop s'ha instal·lat l'aplicació en la que serà la seva ubicació. S'ha provat les funcionalitats bàsiques de generar un curs, associar-li professors i alumnes, crear una granja i fer simulacions. Aquestes proves han donat un resultat similar a les mateixes proves fetes a l'aplicació web en l'entorn de desenvolupament. Mantenint una relació de temps de resposta similar. S'ha provat d'accedir des de Internet a l'aplicació sense que es detectes cap problema.

7. CONCLUSIONS

Un cop acabades la implementació i les proves es fa balanç de quina ha estat la planificació real del projecte, es comenten les desviacions amb la planificació efectuada després de la fase de disseny. Es fa una valoració global de l'experiència que ha suposat la realització del projecte. Al final d'aquestes conclusions es comenten possibles millores i el futur que pot tenir el projecte.

7.1. SEGUIMENT DEL PROJECTE

El seguiment del projecte permet controlar, en la mesura que es van completant les tasques, en quin moment es troba el desenvolupament del projecte. També és important a l'hora de valorar la planificació feta al final de la fase de disseny, mostrant totes les tasques amb la durada real en relació a la que s'havia planificat. D'aquesta manera es fàcil detectar en quines fases el projecte ha patit retards.

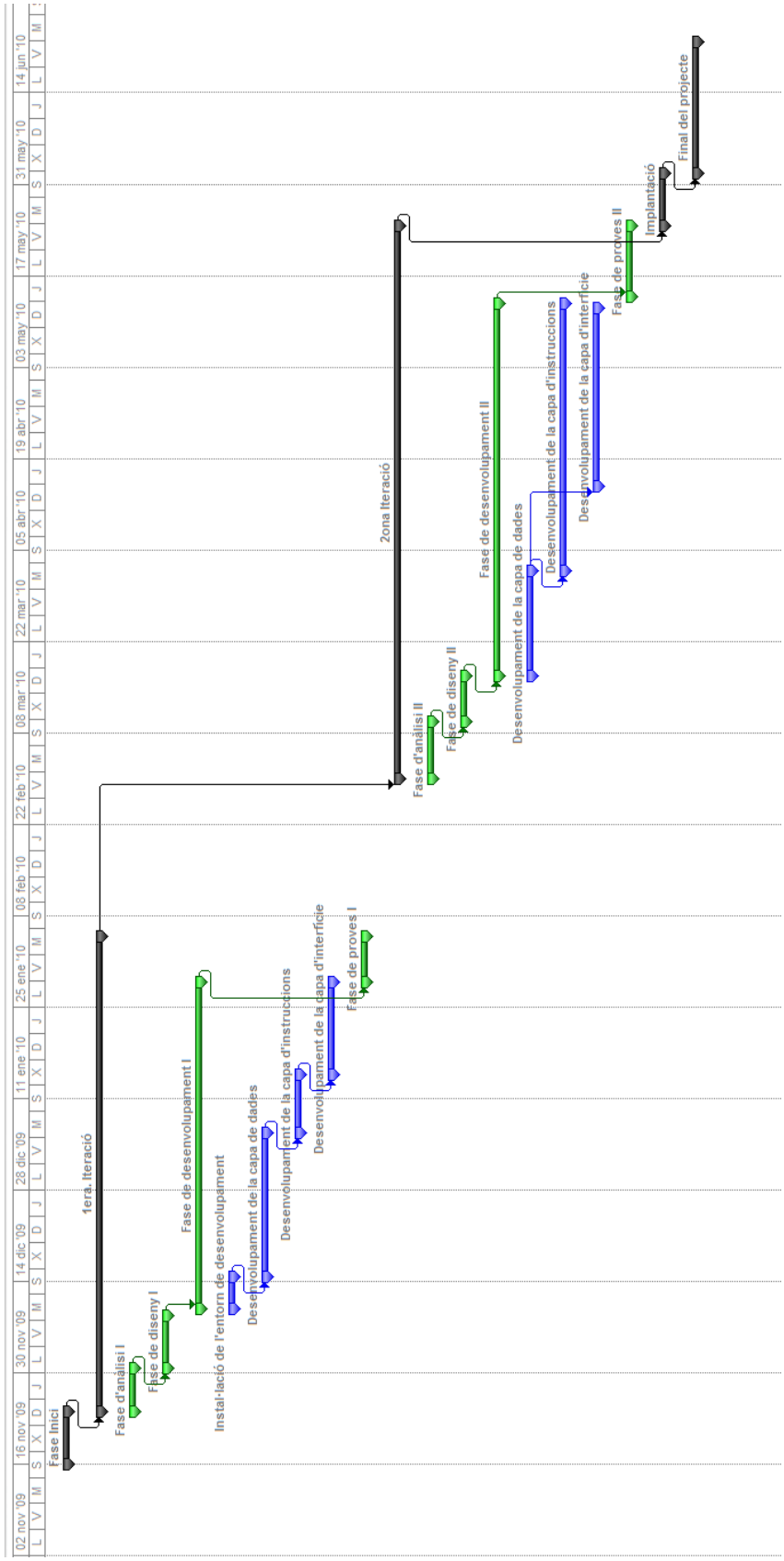
Tasques del projecte

Nom de la Tasca	Durada prevista	Durada real	Inici	Fi
Fase Inici	**5d**	6d	**16/11/2009**	**23/11/2009**
Inici del projecte: Assignació i matriculació del projecte	2h	2h	16/11/2009	16/11/2009
Estudi de requisits des del punt de vista del usuari	2h	2h	16/11/2009	16/11/2009
Estudi de viabilitat	20h	25h	16/11/2009	23/11/2009
Aprovació d'estudi de viabilitat	1h	1h	23/11/2009	23/11/2009
1era. Iteració	**41d**	**52,8d**	**24/11/2009**	**04/02/2010**
Fase d'anàlisi I	**3,4d**	**4,6d**	**24/11/2009**	**30/11/2009**
Anàlisi de requeriments funcionals	6h	10h	24/11/2009	25/11/2009
Casos d'ús	4h	4h	26/11/2009	26/11/2009
Estructura de dades	3h	4h	26/11/2009	27/11/2009
Anàlisi de requeriments no funcionals	1h	1h	27/11/2009	27/11/2009
Estudi de l'arquitectura de l'aplicació i marc tecnològic	3h	4h	27/11/2009	30/11/2009
Fase de disseny I	**7,2d**	**6d**	**30/11/2009**	**08/12/2009**
Estudi del llenguatge de desenvolupament	8h	8h	30/11/2009	02/12/2009
Definició de perfils d'usuari	2h	2h	02/12/2009	02/12/2009
Disseny de la interfície	4h	8h	02/12/2009	04/12/2009
Disseny de la capa de instruccions	8h	8h	03/12/2009	04/12/2009
Disseny de la capa de dades	2,8d	1,6d	07/12/2009	08/12/2009
Disseny de la vaca	8h	4h	07/12/2009	07/12/2009
Disseny de la granja	2h	2h	07/12/2009	08/12/2009
Disseny de l'escenari de simulació	2h	1h	08/12/2009	08/12/2009
Disseny entitat relació de la base de dades	2h	1h	08/12/2009	08/12/2009
Fase de desenvolupament I	**27,6d**	**35,8d**	**09/12/2009**	**28/01/2010**
Instal·lació de l'entorn de desenvolupament	1,4d	2,6d	09/12/2009	14/12/2009
Instal·lació de màquina virtual de Java	1h	1h	09/12/2009	09/12/2009
Instal·lació de servidor web, Apache Tomcat	2h	4h	10/12/2009	10/12/2009

Nom de la Tasca	Durada prevista	Durada real	Inici	Fi
Instal·lació i creació de base de dades MySQL	4h	8h	10/12/2009	14/12/2009
Desenvolupament de la capa de dades	12d	16d	14/12/2009	05/01/2010
Desenvolupament de la classe vaca	50h	55h	14/12/2009	29/12/2009
Desenvolupament de la classe granja	10h	25h	29/12/2009	05/01/2010
Desenvolupament de la capa d'instruccions	4,8d	6,8d	05/01/2010	14/01/2010
Desenvolupament de la simulació d'un dia de la granja	10h	10h	05/01/2010	07/01/2010
Creació de granges	10h	20h	07/01/2010	13/01/2010
Control d'accés d'usuaris	4h	4h	13/01/2010	14/01/2010
Desenvolupament de la capa d'interfície	9,4d	10,4d	14/01/2010	28/01/2010
Maquetat HTML de les interfícies	5h	10h	14/01/2010	18/01/2010
Creació de la pàgina d'estils	5h	5h	18/01/2010	19/01/2010
Interfícies d'accés a l'aplicació	2h	2h	19/01/2010	19/01/2010
Interfícies d'estat de les vaques d'una granja	10h	10h	19/01/2010	21/01/2010
Gràfiques d'evolució de la producció	20h	20h	21/01/2010	27/01/2010
Càrrega dinàmica afegint funcionalitats Javascript	5h	5h	27/01/2010	28/01/2010
Fase de proves I	**2,8d**	**5,2d**	**28/01/2010**	**04/02/2010**
Proves de la capa de dades	2h	5h	28/01/2010	29/01/2010
Proves de la capa d'interfície	5h	5h	29/01/2010	01/02/2010
Proves funcionals	2h	8h	01/02/2010	03/02/2010
Proves d'estrès	5h	8h	03/02/2010	04/02/2010
2ona Iteració	**50,6d**	**60,2d**	**01/03/2010**	**24/05/2010**
Fase d'anàlisi II	**5,2d**	**6,2d**	**01/03/2010**	**09/03/2010**
Anàlisi de nous requeriments funcionals	12h	12h	01/03/2010	03/03/2010
Casos d'ús dels nous requeriments	4h	4h	03/03/2010	04/03/2010
Estructura de dades d'acord amb els nous requeriments	10h	15h	04/03/2010	09/03/2010
Fase de disseny II	**4,8d**	**5,4d**	**09/03/2010**	**22/03/2010**
Redefinició de perfils d'usuari	1h	2h	09/03/2010	09/03/2010
Redisseny de la interfície	10h	10h	09/03/2010	11/03/2010
Redisseny de la capa de instruccions	10h	10h	11/03/2010	15/03/2010
Redisseny de la capa de dades	0,6d	1d	15/03/2010	22/03/2010
Redisseny de la granja	1h	1h	15/03/2010	17/03/2010
Redisseny de l'escenari de simulació	1h	1h	17/03/2010	19/03/2010
Redisseny entitat relació de la base de dades	1h	3h	19/03/2010	22/03/2010
Fase de desenvolupament II	**37,2d**	**41,2d**	**22/03/2010**	**13/05/2010**
Desenvolupament de la capa de dades	7d	12,2d	22/03/2010	02/04/2010
Desenvolupament del pool de connexions	5h	10h	22/03/2010	24/03/2010
Desenvolupament de la classe vaca i d'accions de la vaca	20h	15h	24/03/2010	29/03/2010
Desenvolupament de la classe granja i d'estat de la granja	5h	10h	29/03/2010	31/03/2010
Desenvolupament de la classe d'entorn del curs	5h	25h	31/03/2010	02/04/2010
Desenvolupament de la capa d'instruccions	30,2d	29d	02/04/2010	12/05/2010
Creació de jerarquia de classes	5h	5h	02/04/2010	05/04/2010
Creació de instruccions de control d'usuaris	2h	2h	05/04/2010	05/04/2010
Desenvolupament de les funcionalitats del alumne	7,6d	7,6d	06/04/2010	14/04/2010
Creació de accions sobre les vaques	10h	10h	06/04/2010	07/04/2010
Revisió de la simulació	18h	18h	08/04/2010	13/04/2010
Exportació de dades de la granja	10h	5h	13/04/2010	14/04/2010
Creació de noves vaques	23h	15h	06/04/2010	08/04/2010
Generació d'historial de successos de les vaques	10h	5h	13/04/2010	14/04/2010
Desenvolupament de les funcionalitats del administrador	3,6d	3,1d	26/04/2010	29/04/2010
Creació de cursos	4h	8h	26/04/2010	28/04/2010
Creació d'usuaris professor	4h	2h	28/04/2010	28/04/2010
Copies de seguretat manuals de la base de dades	10h	3h	28/04/2010	29/04/2010
Desenvolupament de les funcionalitats del professor	10d	10,8d	29/04/2010	12/05/2010
Gestió dels cursos	30h	25h	29/04/2010	06/05/2010
Exportació de dades dels cursos	5h	5h	11/05/2010	12/05/2010
Desenvolupament de la capa d'interfície	20,2d	19,4d	14/04/2010	13/05/2010
Creació d'interfícies d'usuari	8d	4d	14/04/2010	20/04/2010
Estat de la granja i gràfiques de producció	20h	10h	14/04/2010	16/04/2010
Detalls de les vaques	10h	5h	16/04/2010	19/04/2010

Nom de la Tasca	Durada prevista	Durada real	Inici	Fi
Taula d'inseminacions	10h	5h	19/04/2010	20/04/2010
Creació d'interfícies d'administrador	3,2d	10h	20/04/2010	22/04/2010
Creació d'interfícies de professor	3d	5d	06/05/2010	13/05/2010
Detalls dels alumnes	15h	25h	06/05/2010	13/05/2010
Fase de proves II	**3,4d**	**6,4d**	**13/05/2010**	**24/05/2010**
Proves de la capa de dades	2h	2h	13/05/2010	14/05/2010
Proves de la capa d'interfície	5h	10h	14/05/2010	18/05/2010
Proves funcionals	5h	5h	18/05/2010	19/05/2010
Proves d'estrès	5h	15h	19/05/2010	24/05/2010
Implantació	**5d**	**6d**	**24/05/2010**	**01/06/2010**
Instal·lació de l'aplicació	5h	10h	24/05/2010	26/05/2010
Proves reals	5h	5h	26/05/2010	27/05/2010
Documentació Instal·lació i manuals d'usuari	15h	15h	27/05/2010	01/06/2010
Final del projecte	**7,4d**	**14,6d**	**01/06/2010**	**21/06/2010**
Generació de documentació, memòria	30h	60h	01/06/2010	17/06/2010
Tancament del projecte	2h	8h	17/06/2010	18/06/2010
Defensa del projecte	5h	5h	18/06/2010	21/06/2010
PROJECTE	**109d**	**139,9d**	**16/11/2009**	**21/06/2010**

Diagrama de Gantt



7.2. DESVIACIONS

Com es pot observar tant en el llistat de tasques i la seva duració real com en el diagrama de Gantt que descriu la distribució de temps que ha calgut per dur-lo a terme, hi ha hagut una pausa en les activitats relacionades amb el projecte durant bona part del febrer. Això ha estat motivat per dos factors: per una banda la coincidència amb el període d'avaluació de les assignatures del curs, i per l'altra la dificultat per programar la segona reunió amb el professor de la facultat de Veterinària.

Aquesta reunió quan s'ha dut a terme ha estat clau en el projecte perquè en ella s'ha validat la feina feta dins la primera iteració. Una feina destinada a desenvolupar el comportament de les entitats de la simulació, per tant és important que el professor hagi comprovat que el funcionament és l'espera't. Un cop validada la primera versió de l'aplicació s'han afegit el gruix de requeriments funcionals necessaris per la seva utilització com a eina formativa.

Pel que fa a les fases de desenvolupament les desviacions no son importants perquè en la planificació ja es preveia un cost important de recursos degut a la complexitat del projecte i el fet d'adquirir la formació necessària per dur-lo a terme durant la seva realització. Tot i així classe molt extenses com la que defineix l'entorn de la simulació han requerit de molt més temps del planificat.

Desviacions més significatives son les relacionades amb les fases de proves que han obligat a refer parts de la implementació per resoldre problemes de funcionament com es descriu al capítol de proves i aquest fet ha allargat la seva realització.

També ha resultat més costós, del que estava planificat, la redacció de la documentació del projecte degut a la quantitat de parts d'aquest que requerien d'una explicació i l'esforç de sintetitzar-la. Ha calgut, a més, revisar a fons l'estructura i els continguts per endreçar el redactat. Tot plegat ha provocat que aquesta tasca s'hagi allargat el doble del temps planificat.

7.3. EXPERIÈNCIA

La experiència ha estat globalment molt positiva. La principal raó és que els coneixements adquirits durant l'any acadèmic s'han vist reforçats per el desenvolupament del projecte. Els conceptes apresos han estat útils per resoldre els problemes reals que ha plantejat el projecte i aquest fet també ha servit per augmentar la motivació per adquirir uns coneixements que, si hagués calgut buscar-los, hagués estat molt més difícil. Tot i així, durant el desenvolupament, el projecte ha anat per davant dels continguts del curs i aquest fet ha provocat que calgués refer algunes parts que estaven fetes abans de veure els conceptes a l'aula. Aquest fet, però, també ha estimulat que durant tot el projecte s'analitzés la feina feta amb més rigor. Sabent que el que ja estava fet es podria haver fet millor ha estimulat que en la mesura del que era possible es treballés en millorar parts de la implementació més crítiques i es posés atenció a buscar estructures mal dissenyades.

Conforme ha anat avançant el projecte també s'ha vist que l'aplicació desenvolupada podia assolir els objectius proposats i el fet de que es plantegi el seu ús de cara el curs vinent a la Facultat de Veterinària de la Universitat Autònoma de Barcelona també ha motivat més implicació final per mirar de deixar un codi ben comentat i amb un format el més estàndard possible per facilitar que algú es plantegi ampliar l'aplicació.

Potser el més difícil ha estat acceptar que després de cada reunió amb el professor sortien nous requeriments que no s'havien tingut en compte. Per una banda s'anava enllestint la feina i per l'altre n'apareixia de nova. Es comprensible que el propi projecte evolucioni durant la seva realització i s'ha assumit que la planificació s'allargués fins a la data límit fixada en l'estudi de viabilitat. Aconseguir que l'aplicació sigui útil passa per acceptar que hi ha funcionalitats que per força s'han d'incloure tot i no estar plantejades des del inici.

7.4. FUTUR DEL PROJECTE I POSSIBLES MILLORES

Aquest projecte neix a arrel de la necessitat de trobar una eina per millorar la formació en l'assignatura de producció bovina de la Facultat de Veterinària de la Universitat Autònoma de Barcelona. L'objectiu és ambiciós i el treball descrit en aquesta memòria és el primer pas per construir un complet paquet de simulació del comportament de les vaques. És una eina que té molt de marge per créixer i que pot cobrir aspectes que vagin més enllà de la producció com és el diagnòstic de malalties, una visió de la gestió econòmica d'una granja de producció lletera o la planificació alimentària del bestiar.

Totes aquestes millores es poden anar afegint de forma modular de manera que les futures aportacions es puguin centrar en àmbits amb una especificació més concreta. I fins i tot es pot pensar en mòduls que treballin amb independència del tipus d'animal que és simulat, de manera que es puguin desenvolupar altres simuladors amb altres tipus de bestiar.

Millores de disseny

Com a conseqüència dels coneixements adquirits durant el projecte s'ha detectat que probablement seria millor treure de la estructura de dades que defineix als objectes Vaca i Granja tot el codi que implementa les funcionalitats que permeten simular. D'aquesta manera es pot fer un disseny que permeti a l'aplicació créixer en el nombre de funcionalitats que treballen sobre aquets dos objectes sense tenir que modificar contínuament la seva estructura.

Es pot aprofitar aquest canvi en el disseny per facilitar la inclusió de noves funcionalitats com poden ser: simular malalties, simular dietes alimentaries, simular gestió econòmica de la granja.

Millores d'interfície

L'ús d'arxius d'idiomes per carregar els textos de la interfície fa que sigui molt fàcil afegir nous idiomes. Només cal generar un nou arxiu canviant el nom

perquè el sistema detecti a quin idioma fa referència i copiar-hi tot el contingut d'un altre dels fitxers d'idiomes. La feina que queda finalment és traduir tots els continguts i afegir un enllaç a l'aplicació perquè estigui disponible un nou idioma.

Per altra banda afegir més interfícies derivades de noves funcionalitats també és prou senzill gràcies a classes que ja generen els menús, les capçalera o els peus de les pàgines de l'aplicació.

Finalment també es poden mostrar gràfiques de producció de les vaques que no és limitin a 365 dies. El nombre de dies que es vol mostrar es pot passar com a paràmetre obrint la possibilitat a afegir la funcionalitat de veure una història productiva variable segons el que especifiqui l'usuari.

8. BIBLIOGRAFIA

8.1. LLIBRES

- **Chopra, V., y otros. 2004.** *Apache Tomcat 5.* : Anaya Multimedia/Wrox, 2004.
- **Eckel, B. 2002.** *Piensa en Java.* Prentice Hall, 2002.
- **Gallardo, D., Burnette, E. y McGovern, R. 2003.** *Eclipse in action.* : Manning, 2003.
- **Richardson, W.C., y otros. 2006.** *Java JDK 6.* s.l. : Anaya Multimedia/Wrox, 2006.
- **Zakas, N.C., McPeak, J. y Fawcett, J. 2006.** *Ajax.* s.l. : Anaya Multimedia/Wrox, 2006.

8.2. REFERÈNCIES A INTERNET

- **Apache, Software Foundation. 2010.** Apache Tomcat 6.0. [En línea] 2010. http://tomcat.apache.org/tomcat-6.0-doc/index.html.
- **Casas, Esther. 2008.** DataSource - Base - Confluence. [En línea] 2008. http://amap.cantabria.es/confluence/display/BASE/DataSource.
- **ChuWiki. 2009.** ChuWiki. *Ejemplo sencillo con JFreeChart: Area, Bar y Line*. [En línea] 2009. http://www.chuidang.com/chuwiki/index.php?title=Ejemplo_sencillo_con_JFreeChart_:_Area%2C_Bar_y_Line.
- **Franco, Angel. 2000.** La clase derivada de Thread. [En línea] 2000. http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/threads/subprocesos.htm.
- **González, Saúl. 2010.** Codex Exempla. *Técnicas de accesibilidad: el Javascript no obstrusivo.* [En línea] 2010. codexexempla.org/curso/curso_4_5.php.

- **Hablutzel, Jaime. 2009.** El espacio de jaime. *JFreeChart: Fácil creación de gráficos estadísticos en Java.* [En línea] 2009. http://elespaciodejaime.wordpress.com/tag/jfreechart/.
- **Limited, Object Refinery. 2009.** JFreeChart. [En línea] 2009. http://www.jfree.org/jfreechart/.
- **Nieto, Andrés. 2005.** anieto2k. [En línea] 2005. http://www.anieto2k.com/2005/12/06/primer-contacto-ajax/.
- **Oracle. 2010.** MySQL 5.5 Reference Manual. [Online] 2010. http://dev.mysql.com/doc/refman/5.5/en/.
- **Project, The JQuery. 2010.** Jquery. [En línea] 2010. http://www.jquery.com.
- **Refsnes, Data. 2020.** W3C Schools. [En línea] 2020. http://www.w3schools.com.
- **Sun, Microsystems, Inc. 2004.** Oracle APIs and Documentation on SDN. [En línea] 2004. java.sun.com/j2se/1.5.0/docs/api/.
- **Wilton-Jones, Mark "Tarquin". 2009.** DOM nodes and tree. [En línea] 2009. http://www.howtocreate.co.uk/tutorials/javascript/dombasics.