Time Tracker

Projecte de Disseny de Software, Curs 2016–17
Enginyeria Informàtica, UAB

Joan Serrat

9 de setembre de 2016

1 Objectiu

Imaginem un estudiant d'informàtica que a més a més de cursar algunes assignatures treballa a temps parcial per compte propi desenvolupant webs per a clients seus. Sovint es queixa de que el temps no li arriba per fer–ho tot, i decideix conèixer amb exactitud, quantitativament, a què dedica el temps, quant i quan, per poder organitzar–se millor. També pensa que necessita quelcom que l'ajudi a no oblidar–se de facturar algunes hores que ha treballat i que, com que no recorda per a quin client les va fer ni exactament quantes eren ni quan van ser, no les pot cobrar. Amb una aplicació de *control o seguiment de l'ús del temps*, podria obtenir la informació que necessita, com mostra la figura 1 i també justificar, davant de qui sigui, el temps facturat mostrant–ne l'escandall.

Una aplicació de seguiment o control de l'ús del temps (en anglès, *time tracker*) permet a un usuari introduir el nom i una descripció de les activitats que porta a terme cada dia, enregistrar els períodes de temps en que l'usuari hi treballa i calcular el temps total dedicat a cada una, o bé el temps dins d'una finestra temporal (el mes passat, per exemple).

La forma de fer-ho és, un cop introduïda una activitat, cada vegada que l'usuari hi comença/acaba de treballar, engega/atura un cronòmetre de manera que la aplicació afegeixi un nou interval a aquesta activitat, que conté la data i hora d'inici i final, i mentre va comptant, mostra el temps dedicat. Quan calgui, es podrà saber quin és el temps total sumant la durada de tots els intervals de la activitat. El temps llavors es pot traduir en una quantitat a facturar, si es tractava d'una activitat remunerada, conèixer la distribució percentual del temps entre les activitats entre dues dates etc.

Existeixen nombroses aplicacions de control de l'ús del temps per a ordinador personal en diferents sistemes operatius, aplicacions web i mòbil. Us recomanem que en descarregueu i proveu algunes per tal d'entendre millor les seves funcionalitats. Per exemple,

- Working Time Tracker, `http://www.allnetic.com`
- GnoTime, per Linux, `http://gttr.sourceforge.net`
- VeriTime Time tracker, `http://www.pcfworks.com`
- myHours.com, aplicació web a `http://myhours.com`

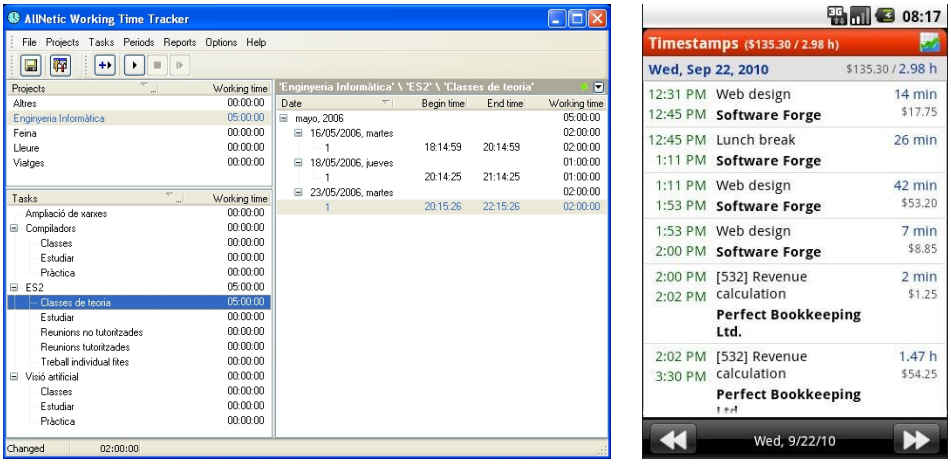


Figura 1: Captures de pantalla de la interfície gràfica del time tracker de AllNetic i Xpert-Timer.

- TSheets, Xpert-Timer i molts altres per Android, altres que podeu trobar a `https://play.google.com/store/search?q=time%20tracker&c=apps`

Com haureu imaginat, l'objectiu és desenvolupar una aplicació de seguiment del temps. En la dinàmica dels projectes reals, rarament us trobareu en començar un projecte amb la llista completa dels requeriments funcionals i no funcionals, cada un d'ells explicat en detall, sense ambigüitats. I sobre tot, *mai* una llista definitiva. Tampoc es fa el disseny, la implementació i la prova de tots els requeriments en paral·lel.

Per aquests motius, hem dividit el projecte en tres fites que contenen la funcionalitat bàsica, per construir el que es diu el *mínim producte viable*. Es clar que es podrien fer moltes millores, extensions i implementar funcionalitats addicionals que són necessàries en tot time tracker, però això ho deixaríem per futures iteracions del projecte. En aquest sentit, l'enunciat fa el paper d'anàlisi de requeriments. Heu de fer el disseny, la implementació i prova del que s'especifiqui en cada fita, però tenint en compte el que trobareu a les següents.

2 Fita 1 : funcionalitat bàsica

Al més alt nivell hi ha *projectes*, entitats que li serveixen a l'usuari per organitzar les tasques en que treballa. No es pot comptar el temps dels projectes directament, sinó només de les seves *tasques*. Els projectes són, doncs, un conjunt de tasques i només de les tasques en comptem el temps. Ara bé, dins d'un projecte hi poden haver tasques i/o més projectes (subprojectes, doncs), els darrers dels quals, al seu torn, poden tenir tasques i/o subprojectes etc. Així, l'usuari pot agrupar jeràrquicament les seves activitats segons el criteri que més s'estimi. L'estudiant d'abans hagués pogut organitzar les activitats en tres projectes diferents : *Feina*, *Formació* i *Lleure*, per exemple. I dins del projecte *Formació* hi podria tenir els subprojectes *Disseny de Software*, *Prova del software*, *Treball final de grau* etc. Dins de cada assignatura, tindria les tasques *Classes de teoria*, *Sessions de pràctiques* etc. De cada tasca la aplicació n'enregistra els seus intervals, és a dir, la data i hora d'inici i final d'un temps dedicat a acomplir aquella tasca. El temps total d'una tasca serà igual doncs la suma de les durades de tots els seus intervals. I el temps total d'un projecte arrel o d'un subprojecte és igual a la suma del temps de les seves

tasques més el dels seus subprojectes, si en té. La jerarquització de projectes permet així tenir tant una visió global com de detall.

Cada cop que es cronometra una tasca (s'engega i al cap d'una estona s'atura el comptador de temps), desem en un interval la data i hora d'inici i de final, així com la durada. Per tant, cal que hi hagi alguna mena de rellotge en el nostre sistema que ens doni aquestes dades.

L'usuari ha de poder especificar quina és la durada mínima d'un interval per a les tasques, és a dir, la unitat de recompte de temps, i que serà un nombre enter i positiu en alguna unitat de temps. Per agilitzar les proves, ara farem que sigui 1 segon, però cal que l'usuari pugui canviar aquesta quantitat si vol, com una opció de configuració. Si un interval dura menys que aquesta quantitat, no es considera com a tal i el temps no es comptarà, com si no hagués passat res. La unitat de recompte de temps serà la mateixa per a totes les tasques. Per un altre cantó, hi ha la freqüència de refresc de la interfície d'usuari, és a dir, cada quant temps es mostren les dades actualitzades de tasques, projectes etc. Per simplificar, farem que coincideixi amb la unitat de recompte de temps.

Com ja hem comentat, cal poder saber el temps total de cada tasca i projecte, al nivell que sigui de la jerarquia. I quan l'usuari estigui cronometrant un tasca, ha de poder observar *mentrestant* (i no nomes quan ha acabat) en la interfície d'usuari com es va incrementant la durada del nou interval i també de la tasca corresponent i de tots els projectes que la contenen. En aquesta fita la interfase d'usuari és simplement la sortida a consola, com expliquem més avall.

L'usuari ha de poder afegir projectes i tasques. En una iteració futura, implementaríem la possibilitat d'afegir manualment intervals (per si ens l'usuari s'oblida d'engegar el cronòmetre), d'eliminar i editar manualment les dades de tasques, projectes i intervals. Tot això, mantenint la consistència amb els antecessors. Però per ara no ho farem ja que no ho considerem part de la funcionalitat bàsica.

Cada projecte i tasca te un nom i, optativament, una descripció (nota explicativa). Un interval, no.

Tal com les hem descrit, totes les tasques són iguals. En realitat no serà així. Hi ha tasques bàsiques, que corresponen a les tasques tal com les hem explicat fins ara. Però en el moment de crear-ne una, l'usuari pot especificar que tingui zero, una o més de les següents característiques:

1. un temps límit donat (per exemple 1 hora) de manera que, quan s'inicia, no es deixa que l'interval corresponent superi aquesta durada per que si ho fa s'atura, la tasca "es para" automàticament.

2. una tasca preprogramada, que s'inicia automàticament en una hora donada si la aplicació està en marxa, per exemple la tasca "dinar" a la una del migdia, cada dia. D'aquesta manera l'usuari s'estalvia d'interaccionar amb la aplicació per introduir tasques que es fan de manera periòdica. I encara més si es combina amb la característica anterior. S'inicien, és clar, si no han estat ja iniciades.

Notem una tasca pot tenir doncs qualsevol combinació d'aquestes capacitats, de les 4 possibles, de cap a totes: una tasca pot ser bàsica, de temps limitat, pre-programada o encara pre-programada i de temps limitat. El disseny ha de preveure que en una futura iteració hi hagi encara més tipus de tasques que si l'usuari vol es poden combinar amb aquestes dues, com per exemple una tasca que hagi de tenir una durada mínima especificada com ara 5 minuts, i

si no s'esborra l'interval corresponent. Llavors tindríem no 4 sinó 8 possibilitats (podem o no tenir cadascuna de les 3 opcions).

Per simplicitat, farem que si una tasca te una o més d'aquests capacitats, les tingui de manera permanent encara que el disseny haurà de permetre treure i afegir capacitats a una tasca en temps d'execució, o sigui, quan ho demani l'usuari en crear-ne una.

Fins ara la aplicació descrita no tindria una utilitat real, ja que un cop s'acaba l'execució es perden totes les dades de projectes, tasques etc. Cal afegir la funcionalitat imprescindible de poder desar en un arxiu, en sortir o en qualsevol altre moment, totes aquestes dades. I també llegir–les de l'arxiu en iniciar l'execució. En aquest aspecte, el pla de projecte preveu que aquesta funcionalitat sigui implementada mitjançant un sistema gestor de base de dades encastat (*embedded*) com ara SQLite i que es puguin exportar i importar aquestes dades en xml. No obstant, en aquesta fase inicial farem servir una alternativa ràpida que és emprar el mecanisme de serialització d'objectes de Java.

Fins aquí, hem descrit la funcionalitat mínima de la aplicació. Una funcionalitat addicional és la possibilitat de cronometrar més d'una tasca simultàniament, del mateix o de diferents projectes. Correspondria a aquelles feines que en realitat són una sola però que es poden aprofitar per a assolir objectius diferents. Per exemple, l'estudiant d'abans programa un mòdul que servirà per la web de dos dels seus clients i que vol facturar dos cops.

En aquesta primera fita no farem pas la interfície d'usuari, sinó que simularem la seva interacció mitjançant una classe que anomenarem `Client`. Aquesta classe tindrà un mètode `main` dins del qual s'instanciaran els objectes que calgui, com ara projectes, tasques, s'engegarà i pararà el cronòmetre, es faran esperes etc. Direm que la interacció es troba *hardcoded*, i ens ha de servir per demostrar que el codi fet passa les proba d'acceptació que us darem (com a mínim).

El codi sempre ha d'estar be comentat, i no esperarem a tenir acabat el projecte per a incloure els comentaris. A la següent fita hi farem més èmfasi (vegeu secció 3.2) però de moment ja començarem a fer-ho be que vol dir :

- Escriviu els comentaris pensant que els ha de poder entendre un programador aliè al projecte, o sigui, no assumiu res i intenteu explicar tot el que sigui rellevant.
- Cal triar molt bé els noms de classes, variables i mètodes per tal de que el codi sigui el màxim d'auto-documentat possible.
- Els comentaris no han de reproduir el codi sinó facilitar-ne la comprensió, explicant el *què* més que no pas el *com* : per exemple per comptes de "sumem els valors de la llista i dividim per la seva longitud", dir "calculem el promig".
- La llengua en que escriviu aquests noms ha de ser la mateixa que la dels comentaris, i cal triar-la ara de manera definitiva. Potser el millor es fer-ho en angles però no és obligatori.

2.1 Lliurament i barem d'avaluació

Per tal de lliurar el projecte, l'heu d'exportar a un arxiu de nom `fita1.zip`: en Eclipse seleccioneu el projecte i feu `File` → `Export ...` → `Zip file`.

Barem d'avaluació:

F Alguna de les següents situacions:

- la funcionalitat mínima respecte al recompte de temps de tasques, projectes i intervals no s'ha assolit
- no hi ha un disseny de classes (expressat amb el diagrama UML), o no es correspon amb el codi
- el disseny te problemes greus de cohesió, acoblament o distribució de responsabilitats
- el disseny no fa servir els dos o tres patrons mínims necessaris que cal descobrir
- comentaris inexistents

E Funcionalitat mínima implementada excepte la persistència i les diferents capacitats que es poden afegir a les tasques (només tenen la bàsica). Cal demostrar-la en superar amb èxit com a mínim el cas de prova que us donem a l'apèndix A. El disseny empra bé els dos/tres patrons de disseny mínims necessaris. Però els comentaris són escassos o incomplets o poc informatius.

D E més

- Implementació de la funcionalitat de persistència ("sortir" i "entrar" de la aplicació veient que en tornar-hi a "entrar" ha les mateixes dades que just abans de "sortir").
- El codi està ben comentat

C D més emprar a tot arreu el mecanisme de logging Logback (*Simple Logging Façade for Java*) per comptes de `System.out.println()` per als missatges d'error, debug etc., amb un *logger* a cada classe rellevant i diferents nivells de prioritat. No pels missatges que implementen la demostració que el codi funciona.

B C més cronometrat de més d'una tasca simultàniament, i codi demostratiu.

A B més disseny que suporti les diferents capacitats de las tasques atenent a les consideracions exposades (extensibilitat, afegir/treure capacitats en temps execució). Cal implementar aquest disseny i fer el corresponent codi demostratiu.

Recomanació: el major esforç de disseny i implementació es troba a E i A. Els altres punts són fàcils d'assolir, sobretot el B quan ja heu fet C.

3 Fita 2 : Disseny per contracte, estil de codificació i un nou requeriment

La primera fita us ha permès entrar en contacte amb l'entorn de desenvolupament, potser també el llenguatge Java i iniciar–vos en el disseny orientat a objecte. La segona fita te dos aspectes. Un és aplicar dues 'bones pràctiques':

1. la idea del disseny per contracte a les classes i els seus mètodes
2. aconseguir que el codi i comentaris s'adhereixin a un estil de codificació donat

L'altre és implementar un nou requeriment, la generació d'informes de diferents tipus i formats. Encara que el resultat hagi de ser el mateix, recomanem que apliqueu primer el disseny per contracte i l'estil de codificació (i comentaris) al codi resultat de la fita 1, i que després implementeu el nou requeriment tot continuant aquestes dues activitats.

3.1 Disseny per contracte

Per tal d'aplicar el disseny per contracte als mètodes de les nostres classes cal primer establir els invariants de classe i les pre i post condicions de cada mètode. No obstant, exclourem els mètodes trivials, com per exemple els accessors (*gets*) que simplement retornin el valor d'un atribut sense cap mena de procés. Per simplicitat implementeu les assercions mitjançant la comanda `assert` de Java, que segueix la sintaxis `assert <condició> : <text de missatge>`. Cal activar-les amb un paràmetre de compilació.

3.2 Estil de codi

L'estil de codi escollit és el que aplica el plugin *Checkstyle* que ajuda enormement en feixuga tasca de realitzar les comprovacions de moltes regles d'estil, i mostrar en el nostre codi on i perquè s'incompleixen. Però és clar, no ho pot fer tot, com per exemple detectar comentaris mal fets.

Escolliu noms de classe, mètode, variable, paràmetre, constant que facin el codi el més entenedor (auto-documentat) possible i us estalviareu comentaris. Les condicions i sentències (individuals o blocs) que puguin ser difícils d'entendre també s'han de comentar. Preneu com a mostra els comentaris del projecte Android que hem donat i seguiu les indicacions de les transparències sobre comentaris del tema d'estil.

Cal passar un corrector ortogràfic (*speller*) de l'idioma escollit, que pot ser català, castellà o angles. Es per això que us hem donat uns arxius diccionari de català i castellà que heu d'afegir al projecte Eclipse.

Una última cosa important: cal mantenir els comentaris actualitzats, és a dir, sempre d'acord amb el codi.

3.3 Informes

L'aplicació ha de generar informes. Un informe és un document que conté els projectes i tasques als que s'hi ha dedicat temps, quant de temps per a cada un i fins i tot quan va ser (intervals). I tot això dins d'un cert període, és a dir, entre dues data i hora especificades. Així per exemple podem saber quantes hores hem dedicat a cada projecte remunerat en el darrer més per poder facturar. De moment, cal implementar dos tipus d'informes segons el seu contingut. Preveiem la necessitat en el futur de tipus addicionals d'informe, basats en càlculs diferents com per exemple el cost de tasques i projectes segons el temps dedicat dins del període, o amb estadístiques sobre la distribució del temps en franges horàries. De moment hem de poder generar els dos següents tipus d'informe :

- Breu, que mostra de cada projecte arrel (els que no són subprojectes d'un altre projecte) el temps total i les dates d'inici i final del projecte, a més de les dues dates entre les que es demana el recompte (període) i la data de generació de l'informe.
- Detallat, que afegeix a l'informe breu el temps dedicat, dins del període demanat de l'informe, de tots els subprojectes i tasques, així com les dades dels intervals. Així mateix, mostra de cada projecte, subprojecte, tasca i interval el seu temps d'inici i final (temps d'inici de l'interval més antic i temps final del més recent, però tenint en compte el període en que es demana l'informe).

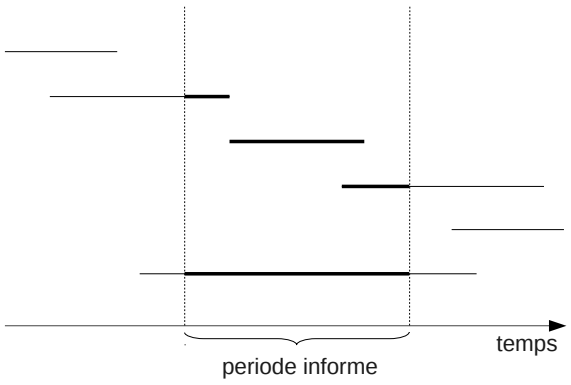


Figura 2: Possibles relacions entre període d'informe i periode de temps de tasca, projecte o interval. El segment de més gruix és el que tindrem en compte a efectes de l'informe.

L'apèndix C mostra un exemple d'informe breu i detallat en format html. En tots dos casos, les dades a mostrar respecte de projectes i eventualment tasques i intervals, resulten d'haver fet prèviament la intersecció dels seus intervals amb el període de l'informe. Al respecte, hi ha les sis situacions possibles que mostra la figura 2.

Els informes s'han de poder generar en diversos formats estàndard i imprimibles. De moment s'han escollit el format text pla (ASCII) i HTML, però més endavant se n'afegiran d'altres com PDF, RTF i XML. Per sort, d'un altre projecte en curs disposem d'una classe `PaginaWeb` que permeten generar el codi HTML de títols (com ara Informe detallat), subtítols (Resum), text del cos del document (S'inclouen en la següent taula ...), línies separadores i taules, que són els elements constituents dels informes HTML esmentats. Aquesta classe la tenim en format d'arxiu JAR i per tant, no la podem modificar a la nostra conveniència. La raó es que ens interessa aprofitar les millores que s'hi vagin incorporant en l'altre projecte, els desenvolupadors del qual ens han promès conservar la interfície (signatura) dels seus mètodes. Una de les operacions d'aquesta classe, `afegeixTaula()`, té com a paràmetre un objecte de classe `Taula` que ja us donem implementada. Es tracta simplement de representar una matriu d'objectes com una llista de llistes d'aquests objectes. Vegeu l'arxiu `Taula.java`.

Al seu torn, la classe `PaginaWeb` es basa en una altre classe `Tag`, que tampoc tocarem però de la que cal incloure el corresponent arxiu JAR en el nostre projecte per poder usar la primera. Disposeu del codi font d'aquestes classes a mode de documentació.

Pel format de text pla, en no haver-hi variacions de tipografia, no ens caldrà cap codi extern. L'apèndix B mostra un exemple d'informe detallat en format text.

En generar un informe, cal efectuar únicament els càlculs necessaris per a ell. O sigui per l'informe breu no hem de fer com si fos el detallat i després descartar el resultat dels càlculs que no ens fan falta. Quan diem càlcul ens referim, d'acord al contingut actual dels informes, a omplir alguna de les taules que apareixen en els informes. En el futur per'o hi podrien haver càlculs que no tinguessin cap relació amb les taules.

3.4 Lliurament i barem d'avaluació

Què cal lliurar ? Igual que a la fita anterior, exporteu el projecte a un arxiu `fita2.zip`.

Barem d'avaluació:

F Algun dels següents factors:

- No se segueix l'estil de codificació.
- Els nous comentaris són inexistents, incomplets o poc informatius.
- El disseny no té en compte els canvis previstos respecte al nou requeriment dels informes.

E Es segueix l'estil de codificació, hi ha suficients i prou bons comentaris i s'ha ampliat el disseny i la implementació per acomodar el nou requeriment dels informes però parcialment: només es generen informes breus i només en un dels dos formats. Tot i així, el seu contingut és correcte. Això vol dir que prèviament s'haurà calculat a ma el resultat esperat i coincideix amb el generat.

D E es generen informes breus correctes en els dos formats. També, el codi ha de seguir segueixi l'estil de codificació, minimitzant el nombre d'incompliments.

C D més els següents ítems :

- El disseny ha de contemplar l'existència dels informes detallats en els dos formats, tot i que les dades que mostren no siguin correctes.
- Implementar la comprovació dels invariants de classe, tot fent un mètode `boolean invariant()` que invocarem allà on convingui.

B C més implementar les pre i post-condicions dels mètodes no trivials només de les classes implicades en la generació d'informes.

A B més generar informes detallats correctes.

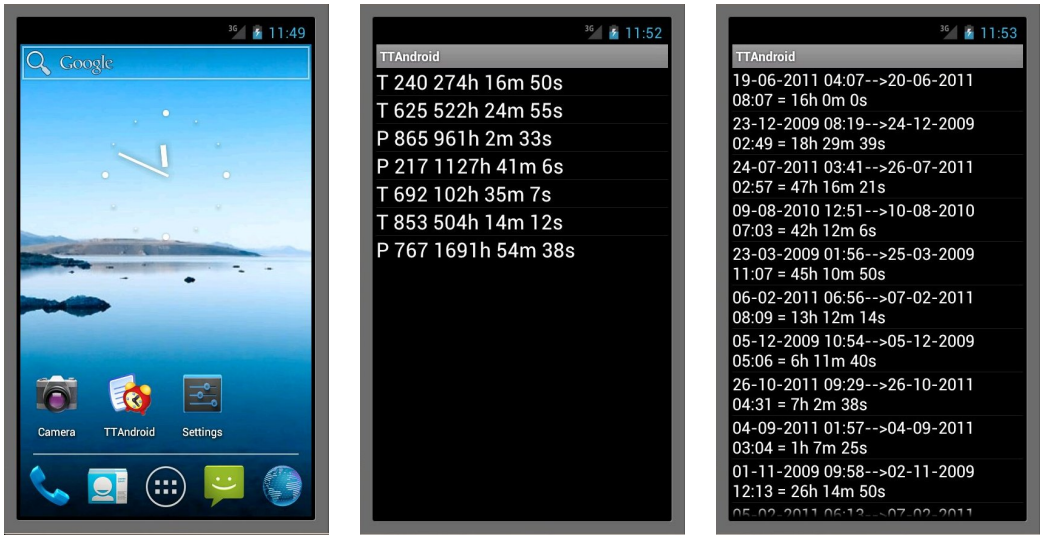
4 Fita 3 : interfície gràfica d'usuari

4.1 Proces

En aquesta fita cal fer el disseny i la implementació parcial de la interfície gràfica d'usuari. Per fer el disseny primer cal conèixer l'estil propi de les interfases Android (què fer i què no, patrons per fer certes coses etc) i, com a part d'això, el catàleg de controls (botons, menús etc.) que l'implementa. Ambdós els podeu trobar molt be explicats al menú que es desplega de la pàgina `https://www.google.com/design/spec/material-design/introduction.html`. Vegeu especialment els ítems *Components* i *Patterns*. En paral·lel, vegeu també les referències bibliogràfiques proporcionades, encara que fan més èmfasi en la programació. Respecte a aquesta, us adrecem especialment a les pestanyes *Training* i *Samples* de `http://developer.android.com/develop/index.html`.

També és una bona idea analitzar la interfície d'aplicacions Android de *time tracking*. Es poden trobar a l'Android Market `https://play.google.com/store/search?q=time%20tracker&c=apps`, i són per exemple TimeSheets, Time Recording, Harvest Time Tracker o Xpert-timer. Vegeu al respecte la darrera llista de problemes.

El segon pas serà pensar el disseny i dibuixar-lo *en paper, a ma, amb llapis (no cal ni regle)*, amb tot el detall possible, així com especificar el lligam entre les diferents vistes davant



(a) (b) (c)

Figura 3: interfície proporcionada de TimeTracker per Android. (a) pantalla *home*, (b) activitats, (c) intervals.

les accions que pugui dur a terme l'usuari. Aquest disseny en paper o *sketchboard* és més fàcil de descartar i refer o modificar, i no genera un vincle "emocional" tant fort com un prototip implementat en codi, que ens resistiríem a canviar perquè ens ha suposat molt més esforç.

Després d'unes quantes iteracions de disseny, en les darreres de les quals podem arribar a fer un disseny d'alta fidelitat, comença la implementació, que te dues parts. La primera és fer codi que en executar-lo mostri vistes, menús etc. en pantalla *i prou*, sense lligar doncs encara aquest codi amb el que implementa les funcionalitats de la aplicació. Així construirem un prototip executable.

La segona és fer aquest lligam, i aquí ens trobem amb un problema de temps, per que cal conèixer el *framework* Android. Això vol dir que és necessari emprar classes com `Activity`, `Service` i altres, de les que cal conèixer el propòsit i l'anomenat cicle de vida. Com que programar la interfície de zero allargaria massa el projecte, us estalvien la part de prova i error i mil detalls, i us donem el codi abastament comentat d'una interfície d'usuari *funcional*: mostra activitats i les seves dates, permet navegar per l'arbre de projectes, tasques i intervals, cronometrar el temps de tasques, i persistència (desar les dades per tornar-les a trobar en executar a aplicació més tard). Es la de la figura 3. Tant podeu desenvolupar modificant i estenent aquest codi, com simplement entendre'l i fer llavors el vostre des de zero.

Com heu d'aprofitar aquest codi ? Haureu vist el primer dia que el TimeTracker "funciona", i això és per que te la llibreria `nucli.jar` que és justament el que demana la fita 1, o sigui, la funcionalitat mínima. Ara heu de substituir aquesta llibreria pel vostre *package* nucli, editant el codi proporcionat per adequar-lo als vostres noms de classe i mètodes. Un cop fet aquest primer pas, ja podeu passar a implementar el disseny que haureu fet de la interfície d'usuari, per que la interfície proporcionada té un munt de problemes d'usabilitat que heu de resoldre. Sense voler ser exhaustius en la seva enumeració, us en indiquem uns quants de força greus:

1. Cal diferenciar què és projecte, tasca i interval. No només per entendre les dades que s'ens mostren, sinó també per saber quines accions hi podem fer. Segur que hi ha maneres

millors que escriure 'P' i 'T' al davant del nom.

2. Cal denotar quin o quins tasques, projectes, interval estan sent cronometrats d'entre els que s'ens mostren en cada moment.

3. Cal algun mecanisme per poder afegir un nou projecte o una nova tasca. Això inclou especificar si es tracta d'una tasca bàsica o si té capacitats afegides (si te una durada limit, si és una tasca de tipus pre-programada), amb les seves dades corresponents. Si no haguéssiu dissenyat/implementat aquestes funcionalitats a la fita 1, no passa res: només caldrà ara poder-ho demanar.

4. L'usuari 'ha de saber' com usar la aplicació, sense haver de llegir instruccions. Tal com algú va senyalar encertadament, només tenim segons per aconseguir que un potencial usuari/client no descarti la nostra aplicació i en provi una altra. Per exemple, ara mateix com podria saber que un *long click* sobre una tasca engega o para el cronòmetre ? En canvi, si que es d'esperar que sàpiga que es pot fet *click* sobre un ítem d'una llista, i que això podria tenir alguna resposta (baixar de nivell en l'arbre). També, que amb el botó o la tecla *back* pujarà de nivell, per que és la forma normal de tornar a la vista anterior.

5. Probablement no cal o no és convenient (es pot?) mostrar alhora totes les dades de tasques, projectes i intervals. Per comptes, s'ha de mostrar per defecte les que es consideren més necessàries i oferir algun mecanisme per visualitzar les altres. Per exemple, si mostrem el text de descripció de tasques i projectes no en veurem gaires simultàniament.

6. El format actual és molt poc atractiu, encara més, les dades resulten difícils de llegir.

7. No es pot sol·licitar un informe, especificant-ne el període, tipus i format

8. Cal algun mecanisme per poder

   (a) eliminar un projecte, tasca o interval

   (b) editar les dades d'un projecte o tasca

   L'edició de tasques i projectes significa canviar alguna de les seves dades. Ara bé, no tindrà sentit canviar la data final, inicial o durada d'un projecte o tasca, ja que depenen de les dels intervals. Per tant ho limitarem a l'edició del nom i descripció.

   Respecte l'eliminació, en tasques, projectes implica eliminar també tots els seus descendents. I en tots els casos (tasques, projectes, intervals) cal actualitzar els antecessors convenientment. Ara be, com que aquesta és una fita de disseny més que d'implementació, només cal fer la interfase com si els efectes de l'eliminació estiguessin implementats, és a dir, que hem de poder-ho demanar però no és obligatori que s'implementi realment, encara que si es fa ho valorarem.

9. Cal denotar el nivell de l'arbre en que ens trobem en cada moment. I si estem 'a dalt de tot' i hi ha tasques que s'estan cronometrant, en prémer *back* can advertir-ho i avisar que se'n deixarà de comptar el temps (es desa l'arbre).

10. L'ordre de la llista de projectes, tasques que es mostra en cada moment hauria de ser tal que en facilités la cerca, i encara millor que l'usuari pogués decidir l'ordre entre diverses opcions. Les més naturals semblen, per tasques i projectes, alfabèticament o de més a menys recent (en ordre de data final decreixent). I pels intervals només aquesta darrera.

11. Estaria bé tenir les opcions de consultar i de parar d'un sol cop el cronòmetre per a totes les tasques actives, evitant així a l'usuari haver-les de buscar per tot l'arbre i parant-les d'una en una. De la mateixa manera, pot ser interessant per l'usuari fer una pausa de totes les tasques actives i després poder-les reprendre totes de cop. Penseu en les accions que li poden interessar a l'usuari, més que en la funcionalitat relacionada amb els mètodes de les classes implementades.

12. La interfície no te cap mena de localització. Caldria poder escollir la llengua i regió, i que això determinés quins missatges i textos apareixeran, així com el format de dates i hores etc., o sigui, tot el relacionat amb i18n i l10n. Com a mínim l'heu de localitzar a dues regions: Catalunya o Espanya, i Estats Units.

4.2 Barem d'avaluació

Pel lliurament, seguiu les instruccions de les fites anteriors (incloent l'informe explicatiu del treball fet). També cal entregar una còpia de l'*sketchboard* de la interfície d'usuari.

Barem d'avaluació:

F Alguna de les següents causes: disseny de la interfície massa pobre, *sketchboard* inexistent, implementació del disseny no existent o molt parcial, que no cobreix els punts per la nota E.

E Resoldre (a nivell de disseny i d'implementació) els punts 1 a 6.

D E més el punt 7.

C D més punts 8 i 9.

B C més punts 10 i 11.

A B més punt 12.

La qualificació, a més del barem, tindrà en compte la qualitat de les solucions i la seva completesa. També valorarem qualsevol millora més enllà dels punts del barem, però ho heu d'acordar prèviament amb el professor de pràctiques.

A Proves fita 1

A continuació us expliquem les proves a fer durant la demostració de la primera fita, per facilitar l'avaluació i també per que pogueu comprovar si la vostra implementació funciona correctament. Pel vostre compte haureu de fer altres proves del mateix estil per tal de cercar possibles errors. Es tracta de que creeu un arbre de projectes i tasques concret, i sobre ell simuleu la interacció amb l'usuari que us proposem, tot mostrant per la consola (pestanya d'Eclipse on es fa l'entrada/sortida textual) com es va fent el recompte del temps.

Dins de la funció `main` de la classe principal de la aplicació de nom `Client`, hi ha d'haver un codi que crei l'arbre de projectes (P) i tasques (T) de la figura 4. Tingueu en compte que P1 podria tenir més endavant projectes germans, fins i tot tasques, depenent de les restriccions que imposeu.

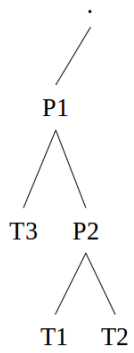


Figura 4: Mini arbre de tasques i projectes per la prova de la fita 1.

A.1 Cronometrat d'una tasca

En cada moment, només cronometrar una tasca. Per tant, des de la funció `main` heu de provar la següent seqüència:

1. Iniciar la aplicació i crear l'estructura de projectes i tasques anterior.
2. Indicar que el temps es comptarà cada dos segons i per tant que la sortida de la aplicació s'actualitzarà o mostrarà també amb aquesta periodicitat.
3. Començar a cronometrar la tasca T3. A partir d'aquest moment s'ha de s'ha de mostrar com es va comptant el temps imprimint cada 2 segons com a les taules 1 a 3. No cal que us hi feu en el format de la taula, només que les columnes estiguin ben indendates per poder llegir-ne el contingut. No cal sobreescriure la taula si no voleu, simplement anar-la reimprimint.
4. Esperar 3 segons i parar el cronòmetre de la tasca T3. Hauríem de veure ara la taula 1.
5. Esperar 7 segons més.
6. Engegar el cronòmetre per la tasca T2.
7. Esperar 10 segons i llavors parar el cronometratge de T2.
8. Cronometrar altre cop T3 durant 2 segons. Hauríem de veure ara la taula 2.

A.2 Cronometrat de més d'una tasca simultàniament

Si el vostre codi permet cronometrar de manera simultània dues o més tasques (del mateix o de diferents projectes), heu de provar a més a més el següent:

1. Iniciar la aplicació i crear l'estructura de projectes i tasques anterior.
2. Indicar que el temps es comptarà cada dos segons i per tant que la sortida de la aplicació s'actualitzarà o mostrarà també amb aquesta periodicitat.
3. Començar a cronometrar la tasca T3.

4. Passats 4 segons, començar a cronometrar la tasca T2.
5. Passats 2 segons, parar el cronometrat de la tasca T3.
6. Passats 2 segons, començar a cronometrar la tasca T1.
7. Passats 4 segons, parar el cronometrat de la tasca T1.
8. Passats 2 segons, parar el cronometrat de la tasca T2.
9. Esperar sense fer res 4 segons i després cronometrar T3 durant 2 segons.

En aquest moment (14 segons després d'haver començat a cronometrar la primera tasca) hauríem de tenir la taula 3.

Nom	Id	Temps inici	Temps final	Durada (hh:mm:ss)
P1	1	5-Set-2016 19:00:00	5-Set-2016 19:00:02	00:00:02
T3	2	5-Set-2016 19:00:00	5-Set-2016 19:00:02	00:00:02
P2	3			00:00:00
T1	4			00:00:00
T2	5			00:00:00

Taula 1: al segons 2 fins a 7 d'haver-se cronometrat una tasca per primer cop.

Nom	Id	Temps inici	Temps final	Durada (hh:mm:ss)
P1	1	5-Set-2016 19:00:00	5-Set-2016 19:00:20	00:00:14
T3	2	5-Set-2016 19:00:00	5-Set-2016 19:00:20	00:00:04
P2	3	5-Set-2016 19:00:10	5-Set-2016 19:00:20	00:00:10
T1	4			00:00:00
T2	5	5-Set-2016 19:00:10	5-Set-2016 19:00:20	00:00:10

Taula 2: al segon 22.

Nom	Id	Temps inici	Temps final	Durada (hh:mm:ss)
P1	1	5-Set-2016 19:10:00	5-Set-2016 19:10:22	00:00:22
T3	2	5-Set-2016 19:10:00	5-Set-2016 19:10:22	00:00:08
P2	3	5-Set-2016 19:10:04	5-Set-2016 19:10:16	00:00:16
T1	4	5-Set-2016 19:10:08	5-Set-2016 19:10:12	00:00:04
T2	5	5-Set-2016 19:10:04	5-Set-2016 19:10:16	00:00:12

Taula 3: cronometrat simultani al segon 22 d'haver-se cronometrat la primera tasca.

B Exemple d'informe detallat en format text

```
-------------------------------------------------------------------------------------------------------
Informe detallat
-------------------------------------------------------------------------------------------------------
Període
 Data
Desde   01/11/2016, 13:05h
Fins a  01/12/2016, 18:30h
Data de generació de l'informe 18/12/2016, 9:00h
-------------------------------------------------------------------------------------------------------
Projectes arrel
No.  Projecte               Data d'inici        Data final          Temps total
1    Pàgina web personal    15/11/2016, 19:00   25/11/2016, 20:00   25h 45m 00s
2    Enginyeria Informàtica 01/11/2016, 9:00    01/12/2016, 19:00   37h 25m 00s
3    Web client Mane SA     01/11/2016, 9:00    01/12/2016, 19:00   37h 25m 00s
-------------------------------------------------------------------------------------------------------
Subprojectes
S'inclouen en la següent taula només els subprojectes que tinguin alguna tasca amb algun interval dins
del període.

No.  Projecte               Data d'inici       Data final          Temps total
2.1  Compiladors            6/11/2016, 12:00   19/12/2016, 20:00   10h 30m 30s
2.2  Disseny de software    3/11/2016, 9:00    27/12/2016, 17:00   25h 20m 20s
2.3  Classes                3/11/2016, 9:00    27/12/2016, 17:00   5h 10m 10s
2.4  Reunions               5/11/2016, 11:00   26/12/2016, 18:00   15h 15m 10s
-------------------------------------------------------------------------------------------------------
Tasques
S'inclouen en la següent taula la durada de totes tasques i el projecte al qual pertanyen.

No.(sub) Projecte  Tasca                  Data d'inici       Data final          Temps total
1                  Programació JavaScript 15/11/2016, 19:00  16/11/2016, 21:10   4h 40m 50s
1                  Repassar enllaços      17/11/2016, 20:10  17/11/2016, 21:25   0h 55m 18s
2.1                Classes de teoria      18/11/2016, 15:00  25/11/2016, 15:00   4h 0m 0s
2.1                Estudiar               27/11/2016, 20:00  27/11/2016, 21:20   1h 20m 30s
2.2                Sessions practiques    12/11/2016, 15:00  19/11/2016, 16:05   2h 8m 22s
-------------------------------------------------------------------------------------------------------
Intervals
S'inclouen en la següent taula el temps d'inici, final i durada de tots els intervals entre la data
inicial i final especificades, i la tasca i projecte al qual pertanyen.

No.(sub) Projecte  Tasca                  Interval  Data d'inici       Data final          Durada
1                  Programació JavaScript 1         15/11/2016, 19:00  15/11/2016, 21:31h  2h 30m 20s
1                  Programació JavaScript 2         16/11/2016, 19:00  16/11/2016, 21:10h  2h 10m 30s
1                  Repassar enllaços      1         17/11/2016, 20:10  17/11/2016, 20:30h  0h 20m 8s
1                  Repassar enllaços      2         17/11/2016, 20:50  17/11/2016, 21:25h  0h 35m 10s
2.1                Classes de teoria      1         18/11/2016, 15:00  18/11/2016, 15:00h  2h 0m 0s
2.1                Classes de teoria      2         25/11/2016, 15:00  25/11/2016, 15:00h  2h 0m 0s
2.1                Estudiar               1         27/11/2016, 20:0h  27/11/2016, 21:20h  1h 20m 30s
-------------------------------------------------------------------------------------------------------
Time Tracker v1.0
```

C Exemples d'informe en format html

Per anar més ràpid hem generat aquests informes manualment i pot ser que les dades no siguin consistents.

Informe breu

Període

	Data
Desde	01/11/2013, 13:05h
Fins a	01/12/2013, 18:30h
Data de generació de l'informe	18/12/2013, 9:00h

Projectes arrel

Projecte	Temps d'inici	Temps final	Temps total
Pàgina web personal	15/11/2013, 19:00h	25/11/2013, 20:00h	25h 45m 00s
Enginyeria Tècnica Informàtica	01/11/2013, 9:00h	01/12/2013, 19:00h	37h 25m 00s
Web client Mane SA	01/11/2013, 9:00h	01/12/2013, 19:00h	37h 25m 00s

---

Time Tracker v1.0

Informe detallat

Període

	Data
Desde	01/11/2013, 13:05h
Fins a	01/12/2013, 18:30h
Data de generació de l'informe	18/12/2013, 9:00h

Projectes arrel

No.	Projecte	Data d'inici	Data final	Temps total
1	Pàgina web personal	15/11/2013, 19:00h	25/11/2013, 20:00h	25h 45m 00s
2	Enginyeria Tècnica Informàtica	01/11/2013, 9:00h	01/12/2013, 19:00h	37h 25m 00s
3	Web client Mane SA	01/11/2013, 9:00h	01/12/2013, 19:00h	37h 25m 00s

Subprojectes

S'inclouen en la següent taula només els subprojectes que tinguin alguna tasca amb algun interval dins del període.

No.	Projecte	Data d'inici	Data final	Temps total
2.1	Compiladors	6/11/2013, 12:00h	19/12/2013, 20:00h	10h 30m 30s
2.2	Enginyeria del software 2	3/11/2013, 9:00h	27/12/2013, 17:00	25h 20m 20s
2.3	Classes	3/11/2013, 9:00h	27/12/2013, 17:00	5h 10m 10s
2.4	Reunions	5/11/2013, 11:00h	26/12/2013, 18:00h	15h 15m 10s

Tasques

S'inclouen en la següent taula la durada de totes tasques i el projecte al qual pertanyen.

No.(sub) Projecte	Tasca	Data d'inici	Data final	Temps total
1	Programació JavaScript	15/11/2013, 19:00	16/11/2013, 21:10h	4h 40m 50s
1	Repassar enllaços	17/11/2013, 20:10h	17/11/2013, 21:25h	0h 55m 18s
2.1	Classes de teoria	18/11/2013, 15:00h	25/11/2013, 15:00h	4h 0m 0s
2.1	Estudiar	27/11/2013, 20:00h	27/11/2013, 21:20h	1h 20m 30s
2.2	Sessions practiques	12/11/2013, 15:00h	19/11/2013, 16:05h	2h 8m 22s

Intervals

S'inclouen en la següent taula el temps d'inici, final i durada de tots els intervals entre la data inicial i final especificades, i la tasca i projecte al qual pertanyen.

No.(sub) Projecte	Tasca	Interval	Data d'inici	Data final	Durada
1	Programació JavaScript	1	15/11/2013, 19:00	15/11/2013, 21:31h	2h 30m 20s
1	Programació JavaScript	2	16/11/2013, 19:00h	16/11/2013, 21:10h	2h 10m 30s
1	Repassar enllaços	1	17/11/2013, 20:10h	17/11/2013, 20:30h	0h 20m 8s
1	Repassar enllaços	2	17/11/2013, 20:50h	17/11/2013, 21:25h	0h 35m 10s
2.1	Classes de teoria	1	18/11/2013, 15:00h	18/11/2013, 15:00h	2h 0m 0s
2.1	Classes de teoria	2	25/11/2013, 15:00h	25/11/2013, 15:00h	2h 0m 0s
2.1	Estudiar	1	27/11/2013, 20:00h	27/11/2013, 21:20h	1h 20m 30s

Time Tracker v1.0