PSP POCKET WATCH

Programmers Manual

All the basic information needed to modify the program

Erick Lares
Erick.lares@gmail.com

Introduction

Time and size estimation are a big part of software development, knowing how much time and effort a software will take is vital to plan the development process and meet deadlines, a client must know when the product they are paying is going to be ready.

To make a good estimation of how much time and effort something will take we can only rely on previous experience either ours or of some other person, but how reliable is this?

To record previous work there several tools at disposal burndown charts, excel sheets, diaries, general applications, etc. But these tools have something in common, they rely on the user inputs, which is subject to errors, things like not recording interruptions, rely on memory alone to fill the record or simply making mistakes can ruin the recording process making that experience useless or inaccurate in the best case.

An interesting study relates to this is “Investigating Data Quality Problems in the PSP” [1], the focus of the study is to expose the data quality problems presented in the Process Software Model [2], one of the main problem in the data is the collection of said data due to mistakes in the user input.

So, the problem at hand is that recording is not accurate enough due to user mistakes and this causes that the experiences we use to make estimations to be tainted with inaccuracy.

One way to solve this would be to create a tool that can record the time and effort put into a project with the less possible user inputs, this tool could be an application that not only record the time put into the project, but also the interruptions and size with almost exact precision, meaning that the user can concentrate on working instead of having the constant need to keep track of the work their doing.

Development Choices

Xamarin: It was selected for two specific reasons, familiarity with C# and flexibility, Xamarin allows the application to run not only in android but also in IOS and Microsoft phones, saving time in having to develop the application for each platform or translate the code to work in all platforms.

Xamarin also allow the use of native code bringing even more flexibility, if any developer want to use android, IOS or Microsoft specific code Xamarin allows.

Model, View, Controller: The implementation of model, view, controller is not 100% correct and the code will need a few fixes before taking full advantage of this software pattern, once such fixes are implemented MVC give the developer the option to change how the user sees the application without affecting how the application work.

Model: Hold the core classes for the application, each one of this classes represent vital information needed by the application, it also holds all the object that need a representation in the database.

Controller: In its current form in work as a bridge between Xamarin and android database with a few extra classes that hold functionality needed by the application, to implement MVC the correct way a lot of the functionality done in view will have to be move here.

View: In its current form hold most of the functionality of the application and how they interact with the user interface.

Classes

Model:

User:

Class Description:
User has no real functionality outside being a possible class that it will be needed for the scalability of the application, if the application is move into and online environment where users can upload and their information to an online server to check their information everywhere the connection between the user and the information is already there.

Class connections:
A user can have one or many projects.
A user can have one set of settings.

Setting:

Class Description:
Settings contains the basic privacy setting for the application, how much info should the application has access to, also how aggressive the application should be with its notifications and finally how much time the application should consider as idle time, currently not implemented.

Class connections:
A setting has one user they belong to.

Project:

Class Description:
A project is the focus of the user, a user will work on a project through it tasks, a user will dedicate time too a project and project object that wraps all that information.
A Project has a name, description, a list of tasks, a list of Sessions and a user that it belongs to.

Class connections:
A project has one user they belong to.
A Project can have one or many tasks.
A Project can have one or many Sessions.

Project Task:

Class Description:
A task is one small functionality or activity that need to be done for a project, all tasks have one estimation of how much time they will need to be complete.

A task has a name, description, a list of Sessions and a project they belong to.

Every sessions that belong to a task also belong to their project.

Class connections:
A task can have one or many Sessions.
A task has one project they belong to.

Session:

Class Description:
A session is the time spend doing a task, a session focus only on one task, the session belongs to the related task as well as to the project of said task.

A session has a start time, end time, a list of interrupts and a list of location where the sessions is being done.

Class connections:
A session can have one or many locations.
A session has one task they belong to.
A session has one project they belong to.
A session can have none or many interrupts

Interrupts:

Class Description:
An interrupt is defined as anything that could cause a work session to temporary stop, currently only tree types of interrupts are implemented:

Changing locations.
Irregular movement detected.
Manual user interrupt.

An interrupt work simply by recording the initial time, end time and reason, then we can subtract this total time to a session to get the real work time.

Changing locations and Irregular movement detected require no user input the application automatically detect and record this kind of interrupts.

An interrupt has a start time, end time and a reason on why the interrupt happened.

Class connections:
An interrupt has a session they belong to.

<u>Location:</u>

Class Description:
Location is the geological information of where the session is being done.

A location has longitude, latitude an address pull from google maps service.

Class connections:
A location has a session they belong to.

<u>Running Info:</u>

Class Description:
Running info is a class that server as a bridge between Xamarin and the information about the application that android provides, more specific if the application is in background and if a notification is needed.

Every time android detects that the application has gone to sleep the background bool will be true, as soon as the application resume, this will value will be false, a notification is needed if the application is in background and the program detect that an interrupt has happened, this will produce a notification.

Class connections:
Has no connections.

Global Utilities:

Class Description:
A utility class that has global variable related to the android devices to keep control on them.

Global utilities have aceelTime the time interval we allow the application to consult the accelerometer, locationTime the time interval we allow the application to consult the GPS.

still, moving, AccelPresiscion, MovementTicks, StillTicks are control values for the accelerometer explained in **View-Session Live**.

Accelerometer is the accelerometer object, LastMovement the last values for any movement in the accelerometer.

Class connections:
Has no connections.

Controller:

Database Controller:

Class Description:
Class that handles all connections to the database, including initialization of the database, creation and dropping of tables and any query require by the application.

SQL:

Class Description:
Interface require by Xamarin and android for the proper initialization of the database [3].

Verify Location:

Class Description:
Public method to verify if a session has change locations, if the location has change the location is save in the database and the change is returned, else the fact that the location hasn’t change is returned.

Visual:

A lot of fixed are needed on this section, there is a lot of redundancy in the methods used though the classes that should be moved into controller or model to take full advantage of MVC, this is due to time constrain and the inexperience of the main developer in Xamarin.

<u>Session Live.xaml/xaml.cs:</u>

Class Description:
This class is the responsible of handling everything that happens when a new session is started as well as giving the user the information they need about the live session.

The information displayed includes:

- Project name.
- Task name.
- Live time.

The actions available to the user include:

- Start a new session.
- Stop the session.
- Pause the session
- Resume the session.

When a new session a chain of event is set in motion, the first thing that happens if that a new session with no end time is placed in the database, at all times there is only one session with no end time in the database, this session is reference though the application as a live session, meaning a session that it’s still ongoing.

After that two threads are fired, one thread to consult the GPS and the other consult the accelerometer.

GPS General Usage:
To read the GPS an infinite looping thread is created, a Boolean is used to allow the reading of the GPS, once inside the method a second Boolean signal every other method that GPS is taking a reading, while the reading is being done the accelerometer is blocked, also the user cannot leave this screen.

Geolocator is used to get the latitude and longitude while Xamarin maps is used give and address to the location [4].

Once the location is obtained the method check is a change occurred, if this was the case an alert will pop up stating that location change was detected, and the user need to resolve this, an interrupt will be created but it will only be stored in the database if the user agrees that the change in location was an interrupt to their work else, it will be discarded.

If the application is in background a request to the OS will be made to push a notification outside the application.

A sleep thread is used to control the frequency of the GPS usage, this can be change in Global Utilities.

This general usage of the GPS is the same in every other screen in the application.

Accelerometer General Usage:
To read the accelerometer a thread is created, this is done so the user interface remains responsive, several failsafe measures are used to prevent the accelerometer to overflow the application with readings given the fact that they happen so fast.

Only one reading can enter the method, a semaphore and deleting the reading event is used to enforce this policy.

Detecting how much movement and for how long is needed to be considered and interrupt is not an exact science, that why there is two buffer, one buffer detects any kind of change in the accelerometer, the other detect every time the accelerometer remain still, if the movement buffer reach his limit an alert will pop up stating that movement was detected and the user need to resolve this and decide if it was an interrupt or not. If the still buffer gets to its limit it will reset both buffers, this is done to ensure that simple movements like moving the device from one table to another is not and interrupt, also this value can be change in global utilities to meet different thresholds.

If movement is detected an interrupt will be created but it will only be stored in the database if the user agrees that the movement was an interrupt to their work else, it will be discarded.

If the application is in background a request to the OS will be made to push a notification outside the application.

This general usage of the accelerometer is the same in every other screen in the application.

Main Page.xaml/xaml.cs:

Class Description:
This class is the first screen the user will see when they start the application, is in charge creating the database.

The information displayed includes:

- List of user projects.
- If session is live.

The actions available to the user include:

- Create a new project.
- Show basic statistics among all projects and tasks.
- View detailed information about a specific project.
- Pause a session (if interrupt is detected).
- Return to live session.

New Project.xaml/xaml.cs:

Class Description:
This class is the screen that handles the creating of a new project, requiring the user to fill every field needed.

The information displayed includes:

- Project name.
- Project description.
- List of tasks to be assigned to the project.

The actions available to the user include:

- Create a new project.
- Take out task out of the list.
- Fill the require fields
- Pause a session (if interrupt is detected).

Project View.xaml/xaml.cs:

Class Description:
This class is the screen that displays all the information related to a specific project.

The information displayed includes:

- Project name.
- Project description.
- Overall time estimation for the entire project.
- Total Time spend working on that project.
- List of tasks that belong to that project.
- Individuals tasks details.
- Delete project (Not implemented).
- Show individual project statistics.

The actions available to the user include:

- Create a new session for a task
- View list of session of the project.
- View list of an individual task.
- Pause a session (if interrupt is detected).
- Return to live session.

Session details.xaml/xaml.cs:

Class Description:
This screen shows the detailed information of a session.

The information displayed includes:

- Session start time.
- Session end Time.
- Task name.
- List of locations.
- List of interrupts.
- Details of selected interrupt.

The actions available to the user include:

- Pause a session (if interrupt is detected).
- Return to live session.

View sessions.xaml/xaml.cs:

Class Description:
This screen shows a list of sessions either from a project or a task.

The information displayed includes:

- List of sessions

The actions available to the user include:

- Pause a session (if interrupt is detected).
- Return to live session.
- Select session to view details.

Native android:

<u>Main activity:</u>

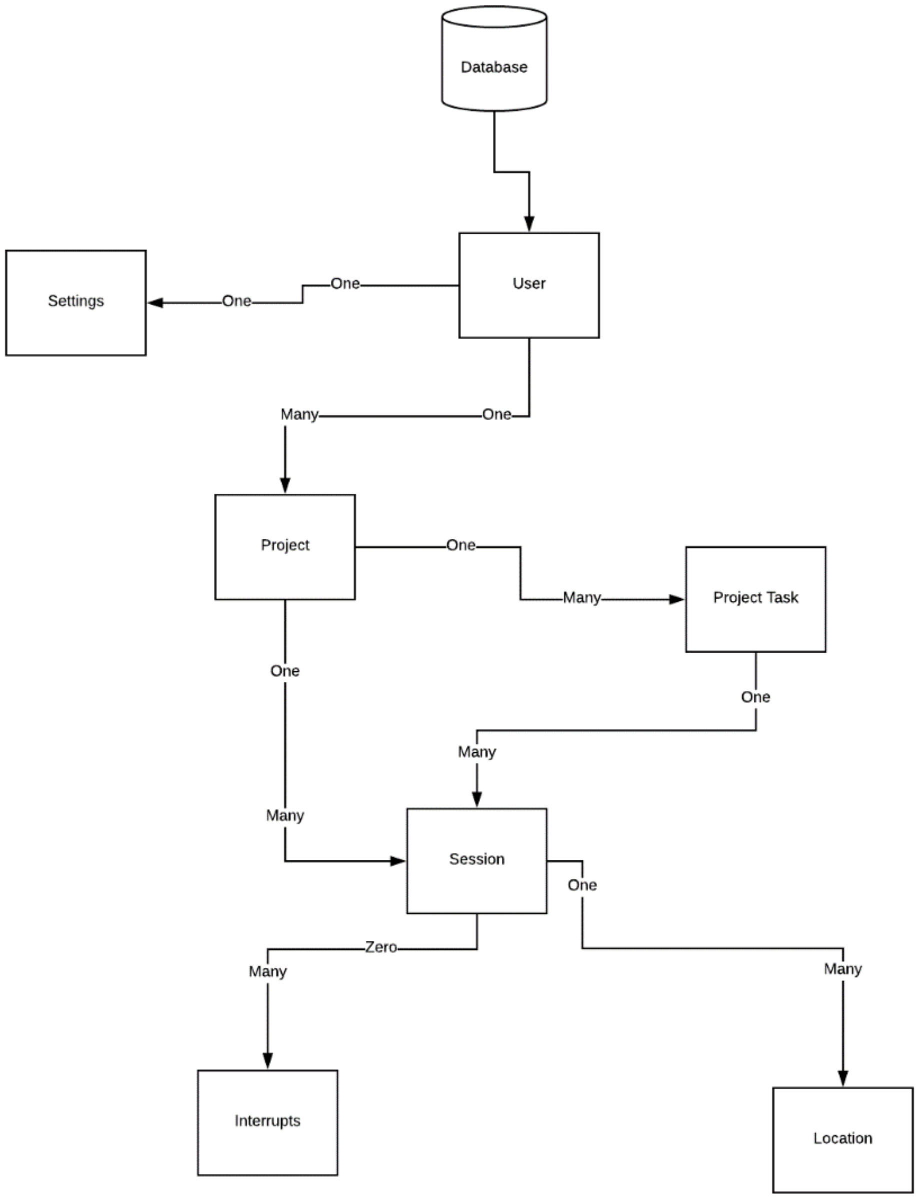
Class Description:
Class that handles the main activity from android, stores in database when the application goes to sleep or resumes functionality, handles pushing notifications outside the application if needed.

<u>SQL Android:</u>

Class Description:
Method require by Xamarin and android for the proper initialization of the database [3].

Database:

The database it’s a simple relational database following the next structure:



Improvements to be made:

There are a lot of thing that can be improve before adding any new features, there is also a lot of new features that could improve the user experience that should be considered before others, this section tries to list them all.

Before adding new Features:

- Improve MVC: As previously said the application currently doesn’t take full advantage of this, before adding any new feature this should be fixed, it will reduce by a lot the size and complexity of the code.
- Add project and task modification, deletes and corrections: The user should have the option to modify a project by adding new task or delete them, delete projects, or even correct sessions times, this was intended but due time constrains was not implemented.
- Clean the code in general.

After the code is better:

- Track user usage of other applications even is no session is live: A huge problem with data recording is user error, no matter how much this application improves on that if the user forgets to use the application in the first place there is no data to store, a counter measure against this is that the application track the user usage of other applications and locations in their day so the user can backtrack and have a reference to what they did during their day to try and remember exactly when they worked on a project.
- Machine learning: Detecting movement and locations is a great feature but one that could bother users, especially if they have a very specific routine, like always moving to a specific location every day to keep working on a session, having to every day tell the application that the change in location is not and interruption can get old really fast, a solution for this is that the application can learn from each project what are the common locations or movements the user does when working on a session, this way familiar locations, movements (even specific applications usage or interactions with persons in the user contacts list) will not trigger interruptions, off course recording all of this and allowing the user to verify and modify this decisions.

Leeson’s learned:

While Xamarin is a great tool and allow a lot of flexibility since not only allow the application to run on all devices but also allow the implementation of native code, this flexibility come with a price.

The documentation for Xamarin for the basic features is great and easy to find, but the deeper and more advance features are used the documentation is harder to find, one example of this is the GPS vs Accelerometer, while the documentation for how to use and implement the GPS is extensive the opposite is true for the accelerometer, the only documentation found was only the library methods, if pure android was used the documentation for every feature is extensive.

Xamarin does great job translating some features to cross-platform, using things like the GPS, basic user interface does not require android native code, but some features are simply not there, one example of this is notifications or finding out if the applications goes to sleep or is resume, to have access to this features a bridge between android and Xamarin had to be build with the use of the database, this means that native android code had to be implemented and if the application needed to run on any other OS then more native code need to be implemented for that OS, which defeat part of the purpose of using Xamarin.

The lesson to be learned here by to anyone that seeks to improve this project is that while Xamarin is a great tool it does not have the same support or documentation that native code has and if cross-platform is something needed Xamarin will help but knowledge on the specific OS is still require to implement some of the of the more advance features.

References:

1. Watts, H. (2000, November). The Personal Software Process. Retrieved April 18, 2017, from http://www.sei.cmu.edu/reports/00tr022.pdf
2. Disney, A. (1998, December). DATA QUALITY PROBLEMS IN THE PERSONAL SOFTWARE PROCESS. Retrieved from https://pdfs.semanticscholar.org/42ca/1ded140ef3492eafce185cb7aa1f2a00191f.pdf
3. Xamarin - Working with Local Databases in Xamarin.Forms Using SQLite. (n.d.). Retrieved from https://msdn.microsoft.com/en-us/magazine/mt736454.aspx
4. Xamarin - Working with Local Databases in Xamarin.Forms Using SQLite. (n.d.). Retrieved from https://msdn.microsoft.com/en-us/magazine/mt736454.aspx