



REPOBLIKAN'I MADAGASIKARA
FITIAVANA - TANINDRAZANA - FANDROSOANA
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE



ECOLE SUPERIEURE POLYTECHNIQUE D'ANTSIRANANA

MENTION MASTER STIC

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME D'INGENIORAT

Electronique et Informatique Industrielles

CALENDRIER POTAGER SOUS ANDROID

Par :

ANDRIAMAMONJISOA FANIRY NAVALONA

Encadreurs :

Dr. ANDRIAMIHARINJAKA HASINA

M. ANDRIANIRINIAIMALAZA TODIZARA

★ Année Universitaire 2015 - 2016 ★



REPOBLIKAN'I MADAGASIKARA
FITIAVANA - TANINDRAZANA - FANDROSOANA
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE



ECOLE SUPERIEURE POLYTECHNIQUE D'ANTSIRANANA

MENTION MASTER STIC

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME D'INGENIORAT

Electronique et Informatique Industrielles

CALENDRIER POTAGER SOUS ANDROID

Par :

ANDRIAMAMONJISOA FANIRY NAVALONA

Encadreurs :

Dr. ANDRIAMIHARINJAKA HASINA

M. ANDRIANIRINIAIMALAZA TODIZARA

Membre de jury :

Dr. ANDRIAMIHARINJAKA Hasina

M. RAZAFINJAKA Nirinarison Jean,

M. ANDRIANIRINIAIMALAZA Philibert,

M. ANDRIANANTENAINA Tsiory Patrick,

Président de jury


Encadreur

Encadreur

Examineur

★ Année Universitaire 2015 - 2016 ★

Promotion : AJeIMa

	<p align="center">ECOLE SUPÉRIEURE POLYTECHNIQUE D'ANTSIRANANA MENTION MASTER STIC B.P. O 201 – ANTSIRANANA – MADAGASCAR Tél.: +261 (0)32 76 395 40 ou +261 (0)32 03 395 40 Courriel : mentionsticespa@ gmail.com</p>
---	---

Maîtriser aujourd'hui la technologie de demain

Mémoire de fin d'étude – Année Académique 2015/2016

01 étudiant en Master 2

Titre : Calendrier potager sous androïde

Objectif

Le calendrier potager est un calendrier de travaux de jardinage qui aide les agriculteurs à mieux récolter sachant que chaque espèce potagère demande des soins spécifiques. Le calendrier vas donc les accompagner mois par mois pour savoir : quelle variété choisir et quand planter et quand récolter, quand effectuer les semis du potager, l'éclaircissage et le repiquage. C'est donc un tableau de bord pour des cultures optimisées. L'objectif de ce travail est donc créer un calendrier interactif de potager.

OBJECTIF : Conception et réalisation d'un calendrier potager sous androïde

Travaux demandés

- Etude bibliographique
- Modélisation de l'application
- Conception et développement de l'application
- Implémentation et test

Encadreur(s)

- ANDRIANAJAINA Todizara
- ANDRIAMIHARINJAKA Hasina , Docteur

Remerciements

Tout d'abord, je tiens à remercier Dieu tout puissant pour m'avoir donné la force et le courage de ne pas abandonner, et aussi l'intelligence et l'espoir pendant la durée du projet.

Et ensuite, j'exprime ma gratitude envers mes encadreurs, Mrs. ANDRIANAJAINA Todizara et Dr. ANDRIAMIHARINJAKA Hasina qui ont bien voulu me donner leurs bonnes volontés, leurs précieux conseils, leur disponibilité malgré leurs charges pédagogiques, ainsi que leurs compétences pour l'élaboration du contenu de ce travail.

Et mes sincères remerciements à toute ma famille, mes amis, mes collègues, et tous ceux qui m'ont soutenu, mais aussi tous ceux qui ont participé, de loin ou de près, à la réalisation de ce projet.

Et enfin, je tiens personnellement à remercier madame la présidente du jury, de m'avoir accordé une chance de présenter ce projet et de m'honorer de sa présence, et à tous les autres membres qui vont porter jugement sur ce travail.

Table des matières

Remerciements	iii
Liste des figures	vi
Liste des tableaux	vi
Introduction	1
1 Théorie sur le calendrier Potager	3
1.1 Définition du potager	4
1.2 Documentation du projet	5
1.3 Rythme de la lune	9
1.4 Calcule des phases lunaires	12
1.5 Les impacte des cycle lunaire dans les cultures	21
1.6 Conclusion	23
2 Android	24
2.1 Description	25
2.2 Le langage Java	28
2.3 Composition de fichier	30
2.4 Environnement de développement	33
2.5 Pattern d'Android	35
2.6 Conclusion	35
3 Implantation du programme	36
3.1 Cycle de vie d'une activité	37
3.2 Les intentes	39
3.3 Modelisation du programme	39
3.4 Présentation de l'application	50
3.5 Conclusion	53
Conclusion	54
Références	ii
A Annexes	I
A.1 CODE SOURCE	I

Liste des figures

1.1	Calendrier potager	6
1.2	Rythme synodique	10
1.3	Rythme sidéral	11
2.1	Logo Android	25
2.2	Couche android	28
2.3	Compilation	30
2.4	MVC Android	35
3.1	Cycle de vie d'une activité	38
3.2	Une intent	39
3.3	Organigramme de calcul de phase de la lune	41
3.4	Diagramme de cas d'utilisation	42
3.5	Diagramme de classes main	43
3.6	Diagramme de classes Model	44
3.7	Diagramme de classe de liste de potager	45
3.8	Diagramme de séquence générale	45
3.9	Diagramme de séquence gestion de nouveau jardin	46
3.10	Diagramme de séquence gestion de jardin enregistrer	46
3.11	Diagramme de séquence calcul du phase lunaire	47
3.12	Diagramme d'état générale	47
3.13	Diagramme d'état gestion nouveau jardin	48
3.14	Diagramme d'état tout le jardin	48
3.15	Diagramme d'état du calendrier	49
3.16	Diagramme d'état de phase lunaire	49
3.17	Splash screem	50
3.18	Ecran d'acceuil	51
3.19	Calendrier	52
3.20	Liste des potages	52
3.21	Liste des potages	53

Liste des tableaux

1.1	La liste de la culture	8
1.2	Les variables de correction	14
1.3	Premier groupe de facteurs de correction	16
1.4	Deuxième groupe de facteurs de correction	18
1.5	troisième groupe de facteurs de correction	19
1.6	Quatrième groupe de facteurs de correction	20

Introduction

Le domaine de l'agriculture a toujours été fortement influencé par des facteurs naturels tels que le climat et les cycles de la nature. Parmi ces influences, les phases lunaires ont depuis longtemps captivé l'attention des agriculteurs en raison de leur prétendu impact sur la croissance des cultures. En réponse à cette fascination, notre mémoire se penche sur le développement d'une application Android révolutionnaire conçue pour les agriculteurs modernes, qui vise à exploiter la sagesse séculaire des phases lunaires pour optimiser leurs calendriers de plantation et de culture. Cette application, baptisée "Calendrier de Potager" se positionne comme un outil novateur pour les agriculteurs soucieux d'améliorer leur rendement et leur durabilité, en harmonie avec les cycles de la nature.

Notre mémoire se consacrera à l'analyse approfondie de cette application, de sa conception à son développement, en passant par son utilité et ses implications pour l'agriculture contemporaine. Nous examinerons également les principaux concepts et théories qui sous-tendent l'influence des phases lunaires sur la croissance des cultures, ainsi que les méthodes scientifiques utilisées pour valider de telles affirmations.

À travers ce travail, nous aspirons à fournir un aperçu complet de l'application "Calendrier de Potager" en mettant en lumière son potentiel pour révolutionner les pratiques agricoles et améliorer la durabilité de l'agriculture. Cette mémoire servira de ressource précieuse pour les agriculteurs, les chercheurs et les passionnés de l'agriculture, en explorant comment la technologie moderne peut s'allier à la sagesse traditionnelle pour favoriser une agriculture plus éclairée et respectueuse de l'environnement.

En même temps, le domaine de la téléphonie mobile évolue constamment depuis son avènement. Ainsi, le sujet propose d'allier ces deux aspects. Il s'agit de construire un calendrier potager pour Android afin de déterminer les cultures d'un jardin pour une date déterminée. Ainsi, nous aborderons différents aspects et concepts pour développer notre application, notamment :

-
- Premièrement, de savoir ce qu'est vraiment une application Android.
 - Deuxièmes on parlera de notre environnement qui est Android.
 - Et enfin, troisièmes on étudiera comment modéliser cette application et de le générer.

Chapitre 1

Théorie sur le calendrier Potager

Sommaire

1.1 Définition du potager	4
1.1.1 Rotation des cultures et disposition des différentes espèces	4
1.2 Documentation du projet	5
1.2.1 Biodynamique	7
1.2.2 Utilisation du calendrier potager	9
1.3 Rythme de la lune	9
1.3.1 Rythme synodique	10
1.3.2 Rythme sidéral	10
1.3.3 Les phases de la Lune	11
1.4 Calcule des phases lunaires	12
1.4.1 Pour la Nouvelle Lune	20
1.4.2 Pour le Premier Quartier	21
1.4.3 Pour la Pleine Lune	21
1.4.4 Pour le Dernier Quartier	21
1.5 Les impacts des cycles lunaires dans les cultures	21
1.6 Conclusion	23

1.1 Définition du potager

Un potager est une parcelle de terrain généralement réservée à la culture de légumes, de fruits, d'herbes aromatiques et parfois de fleurs comestibles, dans le but de fournir une source de nourriture fraîche pour une famille ou une communauté. Les potagers peuvent varier en taille, allant de petits espaces dans les arrière-cours urbaines à de vastes champs ruraux. Le calendrier du potager vise à maximiser le rendement des cultures en les alignant avec les conditions idéales pour leur croissance. Certains calendriers se basent sur des méthodes traditionnelles, tandis que d'autres intègrent des données scientifiques modernes pour fournir des recommandations précises.

Un potager est un espace spécialement aménagé où l'on cultive une variété de légumes, d'herbes aromatiques et parfois de fruits pour la consommation personnelle. Il s'agit d'un jardin productif conçu pour la culture de plantes comestibles, offrant ainsi un approvisionnement de produits frais et locaux pour une alimentation saine et durable. Ce petit coin de terre, soigneusement entretenu, permet de cultiver une diversité de végétaux, favorisant l'autosuffisance alimentaire tout en étant un lieu de connexion avec la nature et d'apprentissage sur la croissance des plantes.

L'objectif global d'un calendrier de potager est d'optimiser la production, la santé et la qualité des cultures tout en respectant les cycles naturels. Cela peut contribuer à une gestion plus durable des ressources et à une meilleure compréhension des interactions entre la nature et l'agriculture.

Il va donc nous accompagner mois par mois pour savoir :

- quelle variété choisir et quand planter ,
- quand effectuer les semis du potager.

C'est donc un tableau de bord pour cultiver votre potager.

1.1.1 Rotation des cultures et disposition des différentes espèces

La rotation des cultures et la disposition des différentes espèces sont des pratiques agricoles visant à optimiser la santé du sol, à prévenir les maladies et les ravageurs, et à améliorer le rendement des cultures. Voici ce que ces pratiques impliquent :

1.1.1.1 Rotation des cultures :

La rotation des cultures consiste à alterner les types de cultures cultivées sur une parcelle de terrain au fil des saisons ou des années. Cette rotation vise à éviter l'épuisement des nutriments spécifiques du sol et à réduire le risque de maladies et de ravageurs qui se développent souvent lorsque les mêmes plantes sont cultivées de manière continue. En alternant les cultures, certaines plantes peuvent même aider à rétablir ou à maintenir l'équilibre des nutriments dans le sol. Par exemple, une culture peut ajouter des nutriments au sol tandis qu'une autre peut en prélever davantage.

1.1.1.2 Disposition des différentes espèces :

La disposition des différentes espèces fait référence à la manière dont différentes plantes sont arrangées dans un jardin ou un champ. Il peut s'agir de la disposition spatiale des cultures ou de la manière dont différentes espèces de plantes sont regroupées. Cette disposition peut avoir des avantages synergiques, tels que la lutte contre les ravageurs, la promotion de la biodiversité, et la maximisation de l'utilisation des ressources. Par exemple, certaines plantes ont des propriétés répulsives pour les insectes nuisibles à d'autres plantes. En les plantant à proximité, on peut réduire le besoin de pesticides. De plus, certaines plantes peuvent avoir des besoins en nutriments différents ou peuvent être complémentaires dans la manière dont elles exploitent les ressources du sol. En résumé, la rotation des cultures et la disposition des différentes espèces sont des pratiques agricoles stratégiques visant à promouvoir la santé du sol, à prévenir les problèmes liés à la monoculture et à améliorer la durabilité globale de l'agriculture.

1.2 Documentation du projet

Dans le cadre de cette mémoire on a demandé l'avis des techniciens d'un groupe spécialisé dans la culture appeler "Union Matanjaka".

L'Union Matanjaka est une organisation paysanne née de l'initiative de leaders paysans du district de Diego II qui œuvre depuis 2003 pour le développement rural des Districts de Diego II et de Diego I.

On nous propose pour simplifier la tâche que pour les deux premières années, les mises en culture proposées par le cycle simple décrit sont regroupées en deux périodes clefs pour chacune des deux années du cycle :

- Grande saison qui se trouve entre le mois Octobre/Novembre jusqu'à Mars/Avril.
c'est une saison chaude favorable à la culture du riz
- Contre saison qui se trouve entre le mois Mars/avril jusqu'en octobre /Novembre

On va parler de la contre saison,elles ont l'avantage de convenir quasiment à tous les climats. Pour la culture on peut donc par exemple suivre cette étape.

- Octobre : préparation du sol (labour, fumure,...)
- Novembre/décembre : semis
- Et après, les pépinières après 2 semaine on peut déjà repiquer/transplanter.
- Entretien : après 3 ou 4 mois on est dans la saison de récolte

Après récolte on reprend le système de culture contre saison.Donc par exemple :

si on a cultivé avant des cultures de la famille des graminées pour entretenir et garder la fraîcheur et les minéraux du sol on cultive les cultures de la famille des légumineuses. On recommence à semer on change de famille.

Ou encore on a,si on cultive des pomme de terre on ne peut pour la culture suivante, cultiver des tomate car il appartienne tous dans la famille de Solanacées, tout ceci est dans un régime ou on a constamment de l'eau. Les pommes de terre ne peut germer a + de 28°C.

Voici une représentation du calendrier de potager Le calendrier définit les potages on 4

Lu	1		↓
Ma	2		↓
Me	3		↓
Je	4		↓
Ve	5		↓

FIGURE 1.1 – Calendrier potager

grande famille.

- fruitier : pomme
- en raciner : pomme de terre
- fleuri : choux fleur
- feuiller : salade

Il existe plusieurs bande dans le calendrier.

- bande rouge : entretien/repiquage

- bande blanche : semi ex : riz tomate, (grain)
- Zone colore : interdit semi

1.2.1 Biodynamique

La biodynamie est une approche agricole holistique qui considère la ferme comme un organisme autonome, interconnecté avec les forces cosmiques et terrestres. Cette méthode a été développée au début du 20e siècle par le philosophe autrichien Rudolf Steiner. La biodynamie intègre des pratiques agricoles organiques avec des éléments ésotériques et spirituels, et elle accorde une attention particulière aux cycles lunaires et planétaires. Voici comment la biodynamie est liée à la rotation des cultures et à la disposition des différentes espèces :

1.2.1.1 Calendrier biodynamique :

Les praticiens de la biodynamie suivent un calendrier spécifique basé sur les positions des planètes et des étoiles. Ce calendrier influence les activités agricoles, y compris la rotation des cultures. Selon la biodynamie, certaines phases lunaires ou positions planétaires peuvent influencer la croissance des plantes et le développement du sol. Les périodes favorables ou défavorables peuvent être prises en compte pour planifier la plantation, la récolte et d'autres activités agricoles.

1.2.1.2 Préparations biodynamiques :

La biodynamie utilise également des préparations spéciales, souvent élaborées à partir de plantes, de minéraux et de compost, qui sont appliquées au sol ou aux cultures pour renforcer la vitalité de la ferme. Ces préparations sont souvent utilisées en association avec des événements cosmiques spécifiques, tels que les éclipses ou les phases lunaires particulières.

1.2.1.3 Rotation des cultures et dynamique des plantes :

La biodynamie encourage une rotation des cultures qui prend en compte les forces cosmiques et la dynamique spécifique de chaque plante. Certains praticiens peuvent également mettre en œuvre des principes de compagnonnage des plantes, où des plantes

spécifiques sont associées pour favoriser la croissance mutuelle et la santé du jardin.

1.2.1.4 Observation des rythmes naturels :

Les agriculteurs biodynamiques sont encouragés à observer attentivement les rythmes naturels et à ajuster leurs pratiques agricoles en conséquence. Cela inclut non seulement les cycles lunaires, mais aussi d'autres cycles cosmiques et terrestres.

Donc on a dressé un tableau qui retrace la liste de la culture possible à Diego Suarez (Ant-siranana).

TABLEAU 1.1 – La liste de la culture

Désignation	Cycle de végétation	Période Favorable	Période de récolte
Arachide			
Aubergine		Mars-Avril/Aout-septembre	Mois après semi
Betterave	100-130 jours	Septembre-décembre	
Carotte	3-4 mois		70-85 jours après semi
Chou de chine	3 mois		2-3 mois
Chou de brocoli	5-6 mois		
Chou-fleur	6-7mois		6 mois après repiquage
Chou	3-3 ½ mois	Février-Juillet	2 ½ mois après
Concombre	3 mois		3 mois
Courge	4-5 mois		
Courgette	2 mois		
Gingembre	9-10 mois		
Haricot sec			120-130 jours
Haricot vert			50 jours
Laitue	70-90 jours		
Melon			
Navet			
oignons	120-170 jours	Mai - septembre	
Petit pois	70-100 jours	Avril- Aout	
persil	5-6 mois		
poireau			4-6 mois après semis
Piment	5-6 mois		
Pois de cap		Mars-septembre	
Poivron	3-6 mois	Mars – juillet	3 mois après semis
Pomme de terre	5-6 mois		
Radis			
Sojas			
Tomate			3 mois après semis

1.2.2 Utilisation du calendrier potager

Pour utiliser le calendrier il faut savoir le type de fructification de la plante que l'on veut cultiver. Pour choisir la période correspondant.

La conception biodynamique du jardinage et de l'agriculture est basée sur la nécessité de concevoir le domaine agricole ou jardinage comme un organisme vivante.

La plante à la différence de l'animal ou du genre humain est totalement ouvert à son environnement. Les différents rythmes du cosmos comme le rythme du soleil, lune et.... Influence beaucoup les plantes. Mais là la question se pose, comment détecter ces influences de ses rythmes dont on ne fait pas l'expérience directe?

Pour répondre à cette question la chercheuse allemande Maria Thun, à la suite de quelque prédécesseur comme Lily Kolisko et Frantz Rulini, depuis une cinquantaine d'années. Étudiant la croissance de la plante et les moments de semis.

Le calendrier de semis peut aider à choisir des dates favorables ou à éviter des dates défavorables, mais c'est aussi un outil de travail pour s'ouvrir au nombreux rythme cosmique et apprendre à toujours mieux ressentir l'instant présent en développant son sens de l'observation.

La plante vive en total dépendance au rythme des journées, des mois et de l'année. Il ne prend pas seulement en compte les positions de la lune, mais aussi celle des planètes du système solaire et leur configuration. En Afrique, chaque année, la végétation s'éveille pendant la saison des pluies, s'épanouit sous le soleil de l'été et mûrit lors de la saison sèche, qui est généralement en hiver.

1.3 Rythme de la lune

Tout comme le soleil, la lune peut faire beaucoup de bien à votre jardin. En effet, notre satellite imprime son rythme à la nature. Jardiner avec la lune, c'est donc également cultiver ses végétaux en harmonie avec la terre.

Comme nous l'avons dit les plantes dépendent de nombreux facteurs comme la lune. On va donc essayer d'étudier le rythme de son dernier.

1.3.1 Rythme synodique

A un certain moment, le soleil est opposée a la lune par rapport à la terre, l'éclaire en pleine face : c'est la pleine lune.

A d'autre moment, il est on conjonction avec la lune, c'est-à-dire qu'il est derrière la lune vue de la terre. La lune est alors totalement invisible, c'est la nouvelle lune.

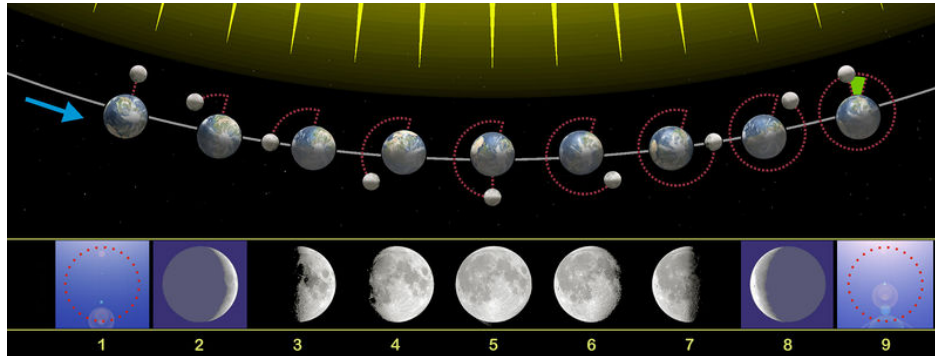


FIGURE 1.2 – Rythme synodique

On a donc une durée de la position de lune croissante à décroissante 29.5 jours avec 15 jours croissante et 15 jours décroissants.

Avec un mouvement de la pleine lune vers la nouvelle lune ou période veille.

1.3.2 Rythme sidéral

Il y aussi le rythme sidéral devant les signes de zodiaque durée 27.3 jours. En effectuant sa rotation autour de la terre, chaque mois, la Lune passe devant les 12 constellations du zodiaque comme le fait le soleil en un an. Le rythme sidéral est la période qui sépare deux passages successifs de la Lune devant le même groupe d'étoiles (d'où l'adjectif sidéral) du zodiaque.

L'utilisation de ce rythme a été popularisée par Maria Thun qui a procédé à des semis journaliers de radis en sol pauvre et sans irrigation. Celle-ci aurait constaté des variations morphologiques, avec des parties de la plante plus ou moins stimulées selon le jour concerné, et donc la constellation du zodiaque du moment du semis.

Inversement les plantes appartiennent à l'une ou l'autre de ces catégories selon la partie de la plante utilisée : il convient alors de les semer et de les soigner préférentiellement aux dates feuille, fruit, fleur ou racine.

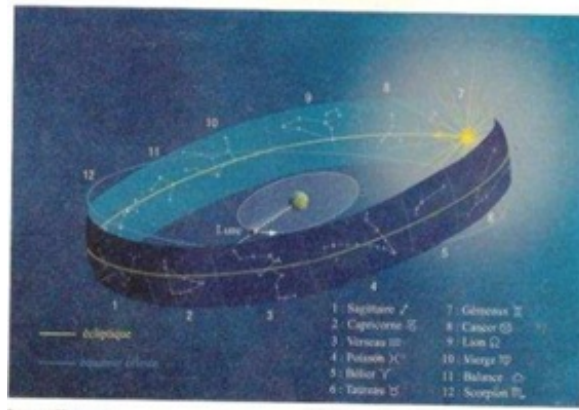


FIGURE 1.3 – Rythme sidéral

1.3.3 Les phases de la Lune

Un cycle lunaire est constitué des phases de la Lune : Nouvelle lune, Premier croissant, Premier quartier, Lune Gibbeuse, Pleine lune, de nouveau Lune Gibbeuse, Dernier quartier, Dernier croissant.

Les phases de la Lune correspondent à ses portions illuminées par le soleil qui sont visibles de la Terre. La Lune tournant en orbite autour de la Terre, ces portions ne cessent de changer en fonction de la position de l'astre.

Un cycle lunaire est aussi appelé une lunaison

Comme nous l'avons dit les plantes dépendent de nombreux facteurs comme la lune. On va donc essayer d'étudier le rythme de son dernier.

Les différentes phases de la Lune ont des noms différents.

Alors que la Lune est croissante, c'est-à-dire que la proportion de sa surface illuminée visible depuis la Terre augmente, et que sa position dans le ciel observé à minuit, parcourt le ciel d'ouest en est en deux semaines, les phases sont :

- la nouvelle lune , la Lune se situe en conjonction avec le Soleil. Elle n'apparaît pas dans le ciel de nuit, mais en journée et présente sa face obscure à la Terre, ce qui la rend difficilement observable ;
- le premier croissant , qui correspond à sa réapparition dans le ciel nocturne ;
- premier quartier , elle est en quadrature et a la forme d'un D ;
- la pleine lune , elle est maintenant en opposition et totalement éclairée par le Soleil.

Si on observe bien on peut observer les mers (ce sont les taches sombres qui sont

en fait des restes de lave qui se sont écoulées sur la Lune).

1.4 Calcule des phases lunaires

Le calendrier dépend entièrement généralement du rythme de la lune. On va donc essayer de savoir les différentes phases. Par définition, les heures de la Maintenant Lune, du Premier Quartier, de la Pleine Lune et du Dernier Quartier sont les heures auxquelles l'excédent de la longitude apparente de la Lune sur la longitude apparente du Soleil est respectivement de 0° , 90° , 180° et 270° .

Ainsi, pour calculer les Instants de ces phases lunaires, il faut calculer les longitudes apparentes de la Lune et du Soleil séparément. (cependant, l'effet de la nutation peut être négligé ici, puisque la nutation, en longitude $\Delta\psi$ n'affectera pas la différence entre les longitudes de la Lune et du Soleil.)

Cependant, si aucune grande précision n'est requise, les instants des phases lunaires peuvent être calculé selon la méthode décrite dans ce chapitre. Les expressions sont basées sur la théorie ELP2000/82 de Chapront pour la Lune (avec des expressions améliorées pour les arguments M , M' , etc., comme mentionné au chapitre 45), et sur la théorie VSOP87 de Bretagnon et Francou pour le Soleil. Les temps résultants seront exprimés en Jours Éphémérides Juliens (JDE), donc en Temps Dynamique.

Cet algorithme calcule la nouvelle Lune du début d'un cycle lunaire. Donc si la nouvelle lune est vers la fin du mois les phases chevaucheront deux mois. Ce qui est le cas pour Avril 2009. Les phases sont en Avril et Mai 2009. Du 25 Avril au 20 Mai. Donc pour certain mois comme 2003-03 le cycle lunaire commence le 2003-04 et non le 2003-03.

Comme on l'a dit on a besoin de deux valeurs d'entrée A pour année et Mo pour mois. Et des divers anomalie M pour trouver les diverses phases lunaires. Pour bien comprendre on va donner la valeur $A=2017$ et $Mo=6$.

On calcul k avec une coefficient 12.3685 une constant pour définir une date simple en siecle Julien qui est :

$$k = (A + mois - 2000) * 12,3685$$

on aura donc :

$$k = 215$$

Une valeur entière de k représente le moment de la Nouvelle Lune. Pour déterminer d'autres phases lunaires, on peut ajouter des fractions spécifiques à k .

0,00 pour la Nouvelle Lune Ex : $k = 215$

0,25 pour le Premier Quartier Ex : $k = 215,25$

0,50 pour la Pleine Lune Ex : $k = 215,5$

0,75 pour le Dernier Quartier Ex : $k = 215,75$

Toute valeur de k autre que celles spécifiquement définies dans ce contexte produira des résultats qui n'ont pas de signification logique ou pratique. En l'occurrence, la valeur est associée à la Nouvelle Lune du 6 janvier 2000. Les valeurs négatives de k sont utilisées pour représenter des phases lunaires antérieures à l'année 2000. Il est important de respecter ces paramètres spécifiés, car des valeurs de k non conformes conduiraient à des conclusions dénuées de sens dans le cadre de cette méthode particulière.

Avec les k obtenu on calcule T qui est le temps en siècles juliens depuis l'époque 2000 ; il est obtenu avec une précision suffisante à partir de :

$$T = k \setminus 1236,85 \text{ on alors } T = 0,17464$$

Ensuite on cherche les anomalies moyennes du Soleil au temps et convertit tous les angles que fait la lune par rapport à la terre dans un intervalle 0° - 360° grâce à la fonction.

Les anomalies ou angles moyens du soleil en fonction du temps est :

$$M = 2.5534 + 29,10535669 * k - 0.00000218 * T^2 - 0,00000011 * T^3$$

$M = 6289,310444$ Les anomalies ou angles moyens de la Lune :

$$M' = 201,5643 + 385,81693528 * k + 0,0107582 * T^2 + 0,00001238 * T^3 - 0,000000058 * T^4$$

$M' = 83538,0226$ L'argument de la latitude de la Lune :

$$F = 160,7108 + 390,67050284 * k - 0,0016118 * T^2 - 0,00000227 * T^3 + 0,00000001 * T^4$$

$F = 84545,5393$ Longitude du nœud ascendant de l'orbite lunaire :

$$\Omega = 124,7746 - 1,56375588 * k + 0,0020672 * T^2 + 0,00000215 * T^2$$

$$\Omega = -212,9965897$$

Nous utilisons un total de 14 variables, chacune correspondant à une phase spécifique des cycles lunaires. Chacune de ces variables est soigneusement définie et représente une mesure précise de l'état de la Lune à un moment donné.

TABLEAU 1.2 – Les variables de correction

	14 Variables de correction	NL en[°]	PL en[°]	PQ en[°]	DQ en[°]
A ₁	$299,77 + 0,107408 * k - 0,009173T2$	322,9698482	322,916145536444	322,889294182478	322,942996871906
A ₂	$251,88 + 0,016321 * k$	255,405336	255,3971755	255,39309525	255,40125575
A ₃	$251,83 + 26,651886 * k$	6008,63737	5995,311433	5988,6484615	6001,9744045
A ₄	$349,42 + 36,412478 * k$	8214,515248	8196,309009	8187,2058895	8205,4121285
A ₅	$84,66 + 18,206239 * k$	4017,207624	4008,1045045	4003,55294475	4012,65606425
A ₆	$141,74 + 53,303771 * k$	11655,35454	11628,7026505	11615,37670775	11642,02859325
A ₇	$207,14 + 2,453732 * k$	737,146112	735,919246	735,305813	736,532679
A ₈	$154,84 + 7,306860 * k$	1733,12176	1729,46833	1727,641615	1731,295045
A ₉	$34,52 + 27,261239 * k$	5922,947624	5909,3170045	5902,50169475	5916,13231425
A ₁₀	$207,19 + 0,121824 * k$	233,503984	233,443072	233,412616	233,473528
A ₁₁	$291,34 + 1,844379 * k$	689,725864	688,8036745	688,34257975	689,26476925
A ₁₂	$161,72 + 24,198154 * k$	5388,521264	5376,422187	5370,3726485	5382,4717255
A ₁₃	$239,56 + 25,513099 * k$	5750,389384	5737,6328345	5731,25455975	5744,01110925
A ₁₄	$331,55 + 3,592518 * k$	1107,533888	1105,737629	1104,8394995	1106,6357585

Les équations sont fournies par Jean Meeus dans le chapitre 47. Cependant, chaque variable est associée à des facteurs de correction visant à compenser les erreurs. L'équation adopte la forme $x * \sin(variable)$, où les fonctions trigonométriques sont exprimées en radians.

Les groupes de correction se répartissent comme suit : Le premier groupe de facteurs de correction s'applique à toutes les phases lunaires. Pour chacune de ces phases, des facteurs de correction spécifiques sont pris en compte.

TABLEAU 1.3 – Premier groupe de facteurs de correction

Équation	résultat pour NL	résultat pour PL	résultat pour PQ	résultat pour DQ
$+0,000325x \sin A_1$	-0,000195726446451076	-0,000195969543384474	-0,000196091027333202	-0,000195848016479172
$+0,000165 * \sin A_2$	-0,000164911	-0,00015966996260938	-0,000159666999759314	-0,000159672924649695
$+0,000164 * \sin A_3$	0,000124787	-0,00013485024979265	-0,000123109904148533	-0,000144768995875441
$0,000126 * \sin A_4$	-0,000114641210533772	-0,000125236904802144	0,00441993714893402	-0,000121468935308566
$+0,000110 * \sin A_5$	9,2470254564845E-05	8,1880045261142E-05	7,57927028433038E-05	8,74509416967826E-05
$+0,000062 * \sin A_6$	4,35685049673676E-05	5,87261176018441E-05	6,17272086710506E-05	5,25625792498189E-05
$+0,000060 * \sin A_7$	1,76885674854124E-05	1,64569354783273E-05	1,58382545240628E-05	1,70737300324346E-05
$+0,000056 * \sin A_8$	-5,15016559550923E-05	-5,279824801507E-05	2,86106842219774E-05	-5,21764677534537E-05
$+0,000047 * \sin A_9$	1,37825513215227E-05	2,39835221158898E-05	-5,33663647699318E-05	1,90174169348782E-05
$+0,000042 * \sin A_{10}$	-3,37637251993606E-05	-3,37371492901123E-05	-3,37238470347936E-05	-3,37504420128866E-05
$+0,000040 * \sin A_{11}$	-2,01655130139186E-05	-2,07188860792644E-05	-2,09935687974932E-05	-2,04428615259341E-05
$+0,000037 * \sin A_{12}$	-7,36315704146046E-06	-1,47997836878978E-05	-1,82912047200385E-05	-1,1143527025191E-05
$+0,000035 * \sin A_{13}$	-5,8433001652702E-06	-1,33189169430474E-05	-1,68321648305043E-05	-9,6407839056416E-06
$+0,000023 * \sin A_{14}$	1,0632282711347E-05	9,98776778602614E-06	9,66178942716129E-06	1,03112920451936E-05
Somme S1 de ce groupe :	-0,000563270890046784	-0,000560065256360811	-0,000556294647033336	-0,000562496994576873

Simultanément, une variable E est calculée en tant que facteur d'ajustement. Cette variable est introduite pour prendre en considération la variation de l'excentricité de l'orbite terrestre autour du Soleil, laquelle évolue au fil du temps. E est utilisée pour moduler les termes dans les équations mathématiques qui dépendent de l'angle M, de manière à refléter cette évolution. Cette variable est exprimée en temps en siècles juliens, ce qui peut donner une valeur négative si la date choisie précède l'époque de l'an 2000.//

Ensuite, les angles suivants, qui sont exprimés en degrés, peuvent être ramenés dans la plage de 0 à 360 degrés, voire convertis en radians si nécessaire, avant de continuer le calcul.

$$E = 1 - 0.002516 * T - 0.0000074 * T^2$$

$$E = 0,9995604$$

Il y a les Deuxième groupe de facteurs de correction pour la nouvelle lune.

TABLEAU 1.4 – Deuxième groupe de facteurs de correction

Équation	résultat pour NL
$-0.40720 * \sin M'$	-0,125984792788553
$+0.17241 * E * \sin M$	0,0319658396790933
$+0.01608 * \sin 2M'$	0,0094618684493312
$+0.01039 * \sin 2F$	-0,00982857411426661
$+0.00739 * E * \sin(M' - M)$	-0,00354867148487472
$-0.00514 * E * \sin(M' + M)$	0,000655768234631119
$+0.00208 * E^2 * \sin 2M$	-0,000757571095532711
$-0.00111 * \sin(M' - 2F)$	-0,000887138316079663
$-0.00057 * \sin(M' + 2F)$	0,000569929888533307
$-0.00056 * E * \sin(M' + 2F)$	-0,00023970740970152
$-0.00042 * \sin(3M')$	-0,00034007965122726
$+0.00042 * E * \sin(M + 2F)$	0,000364987973385152
$+0.00038 * E * \sin(M - 2F)$	-0,000375919531781889
$-0.00024 * E * \sin(2M' - M)$	0,000174688720355239
$-0.00017 * \sin \omega$	-9,25801496041047E-05
$-0.00007 * \sin(M' + 2M)$	-9,25801496041047E-05
$+0.00004 * \sin(2M' - 2F)$	4,09834813626744E-06
$+0.00004 * \sin(3M)$	2,29621470871518E-05
$+0.00003 * \sin(M' + M - 2F)$	2,12374095103306E-05
$+0.00003 * \sin(2M' + 2F)$	-2,69051612104721E-05
$-0.00003 * \sin(M' + M + 2F)$	-2,86700959268883E-05
$+0.00003 * \sin(M' - m + 2F)$	-2,93884967540057E-05
$-0.00003 * \sin(M' - m + 2F)$	2,95630486710799E-05
$-0.00002 * \sin(M' - M - 2F)$	1,34774050429358E-05
$-0.00002 * \sin(3M' + M)$	1,37362682155721E-05
$+0.00002 * \sin(M' - m + 2F)$	1,90308785538786E-05
Somme S2 de ce groupe :	-0,0988228098449667

Il y a les troisième groupe de facteurs de correction pour la pleine lune.

TABLEAU 1.5 – troisième groupe de facteurs de correction

Équation	résultat pour PL
$-0.40614 * \sin M'$	0,0362036386594269
$+0.17302 * E * \sin M$	0,0737513507458925
$+0.01614 * \sin 2M'$	0,0028660094764077
$+0.01043 * \sin 2F$	-0,00676102471884065
$+0.00734 * E * \sin(M' - M)$	0,00370784369915148
$-0.00514 * E * \sin(M' + M)$	0,00177143804836564
$+0.00208 * E^2 * \sin 2M$	-0,00161092093055625
$-0.00111 * \sin(M' - 2F)$	0,00064132713449474
$-0.00057 * \sin(M' + 2F)$	-0,000406708644587854
$-0.00056 * E * \sin(M' + 2F)$	0,000145006033813607
$-0.00042 * \sin(3M')$	0,000109830359957422
$+0.00042 * E * \sin(M + 2F)$	-0,000346045946697358
$+0.00038 * E * \sin(M - 2F)$	0,000139207323175545
$-0.00024 * E * \sin(2M' - M)$	-9,06259016408765E-05
$-0.00017 * \sin \omega$	-4,98163290800534E-05
$-0.00007 * \sin(M' + 2M)$	2,01086226272516E-05
$+0.00004 * \sin(2M' - 2F)$	3,87652294652468E-05
$+0.00004 * \sin(3M)$	2,61200086286397E-05
$+0.00003 * \sin(M' + M - 2F)$	-2,31941462811772E-05
$+0.00003 * \sin(2M' + 2F)$	1,03983096350532E-05
$-0.00003 * \sin(2M' + 2F)$	-2,83251891162595E-05
$+0.00003 * \sin(M' + M + 2F)$	1,03983096350532E-05
$-0.00003 * \sin(M' - m + 2F)$	-2,83251891162595E-05
$-0.00002 * \sin(M' - M - 2F)$	-3,49075827834396E-06
$-0.00002 * \sin(3M' + M)$	3,43847774701176E-06
$+0.00002 * \sin 4M'$	6,98999354949552E-06
Somme S3 de ce groupe :	0,110232446969867

Pour le premier et le dernier quartier on utilisera les calcule d'un dernier groupe de facteur de correction lui aussi le calcul de sinus est on radiant.

TABLEAU 1.6 – Quatrième groupe de facteurs de correction

Équation	résultat pour PQ	résultat pour DQ
$-0.62801 * \sin M'$	-0,0283391141830033	-0,112527937813238
$+0.17172 * E * \sin M$	0,121636550937766	0,148636209897223
$-0.01183 * E * \sin M' + M$	-0,0079308772026438	0,0110851201702104
$+0.00862 * \sin 2M'$	0,000777166993994434	-0,00303909963199027
$+0.00804 * \sin 2F$	0,00803688690636709	0,00679852940428807
$+0.00454 * E * \sin(M' - M)$	-0,00333408157339479	0,00342107881634874
$+0.00204 * E^2 * \sin 2M$	-0,00203881184281414	-0,00179228663419781
$-0.00180 * \sin(M' - 2F)$	-0,00179521002561056	-0,00166960231480066
$-0.00070 * \sin(M' + 2F)$	-0,00069989511033489	0,00051537472377116
$-0.00040 * \sin(3M)$	-0,000287474814157902	-1,59700599030213E-05
$-0.00034 * E * \sin 2M' - M$	0,000259842298874262	0,000212031818795981
$+0.00032 * E * \sin(M + 2F)$	-0,000221109082752543	8,3871045378237E-06
$+0.00032 * E * \sin(M - 2F)$	0,00023362863464245	0,000285118156199247
$-0.00028 * E^2 * \sin(M' + 2M)$	0,000279690888791854	-0,000265926328999593
$+0.00027 * E * \sin(2M' + M)$	0,000171739463858217	0,000265731451333956
$-0.00017 * \sin \omega$	0,000164788998930338	0,000164203689512474
$-0.00005 * \sin(M' - M - 2F)$	-3,28914751857463E-05	7,66639729186511E-06
$+0.00004 * \sin(2M' + 2F)$	3,99220197802469E-05	2,41232012075864E-05
$-0.00004 * \sin(M' + M + 2F)$	2,89221960321067E-05	8,19649610471887E-06
$+0.00004 * \sin(M' - 2M)$	-3,99767034625458E-05	3,80092345047314E-05
$+0.00003 * \sin(M' + M - 2F)$	2,28109012750209E-05	-2,38659806469122E-05
$+0.00003 * \sin 3M$	2,15606110618426E-05	1,1977544927266E-06
$+0.00002 * \sin(2M' - 2F)$	-1,98606620348694E-05	-1,95900452994114E-05
$+0.00002 * \sin(M' - M + 2F)$	-1,39741784927232E-05	1,91614384920573E-05
$-0.00002 * \sin(3M' + M)$	-1,20155240153713E-05	1,9999873735738E-05
Somme S4 de ce groupe :	0,0869082184734702	0,0521558608189752

On additionne le tout pour avoir S4

On calcule le jour le plus

$$JDE = 2451550,09766 + 29,53058886 * k + 0,00015437 * T_2 - 0,000000150 * T_3 + 0,00000000073 * T_4$$

$$JDE = 2457928,70485$$

Ainsi on peut calculer les différentes phases lunaires.

1.4.1 Pour la Nouvelle Lune

$$JD = JDE + S1 + S2$$

Si le mois est 01 : NL = JD = 2454858,33081789

Si le mois est > 01 : $NL = JD + (29.530588852) * (M - 1)$

$JD=2457928,6055$

1.4.2 Pour le Premier Quartier

On Augmente K de 0.25

$$JD = JDE + S1 + S4 + W$$

Si le mois est 01 : $PQ = JD$

Si le mois est > 01 : $PQ = JD + (29.530588852) * (M - 1)$

$PQ=2457906,2467$

1.4.3 Pour la Pleine Lune

On Augmente K de 0.25

$$JD = JDE + S1 + S3$$

Si le mois est 01 : $PL = JD$

Si le mois est > 01 : $PL = JD + (29.530588852) * (M - 1)$

$PL=2457914,0492$

1.4.4 Pour le Dernier Quartier

On Augmente K de 0.25

$$JD = JDE + S1 + S4 - W$$

Si le mois est 01 : $DQ = JD$

Si le mois est > 01 : $PL = JD + (29.530588852) * (M - 1)$

$DQ=2457921,7724$

1.5 Les impacte des cycle lunaire dans les cultures

Les phases lunaires ont une longue histoire d'influence sur l'agriculture et la croissance des cultures. Bien que la science moderne ait apporté de nombreuses réponses aux questions sur les cycles de croissance des plantes, l'impact des phases lunaires reste un sujet de discussion et d'étude.

L'influence la plus connue des phases lunaires est liée à la marée. La lune exerce une force gravitationnelle sur la Terre, provoquant des variations dans les niveaux d'eau, ce qui peut avoir un effet sur l'irrigation et le drainage des cultures près des zones côtières.

Certains agriculteurs et jardiniers traditionnels planifient leurs activités agricoles en fonction des phases de la lune. On pense que la lune croissante est propice à la plantation de cultures qui portent des fruits au-dessus du sol, tandis que la lune décroissante est favorable à la plantation de cultures souterraines ou à la récolte. Cependant, ces pratiques sont souvent basées sur des croyances traditionnelles et manquent d'une base scientifique solide.

Plantations et récoltes en fonction des phases lunaires : Selon la croyance traditionnelle, certaines phases lunaires sont considérées comme propices à la plantation, tandis que d'autres sont recommandées pour la récolte. Par exemple, la lune croissante est souvent associée à la plantation de cultures qui portent des fruits au-dessus du sol, tandis que la lune décroissante est associée à la plantation de cultures souterraines et à la récolte.

Influence sur les marées et l'irrigation : La lune exerce une force gravitationnelle sur la Terre, ce qui provoque les marées. Cette force peut également affecter la façon dont l'eau est distribuée dans les sols. Dans les régions côtières, les agriculteurs tiennent souvent compte des phases lunaires pour la planification de l'irrigation et du drainage.

Traditions culturelles et héritage : Les croyances liées au cycle lunaire dans la culture des potagers sont souvent enracinées dans des traditions ancestrales et culturelles. Les connaissances transmises de génération en génération ont contribué à perpétuer ces pratiques.

Recherche scientifique limitée : La science moderne a en grande partie remis en question ces croyances. Les études scientifiques sur l'impact des phases lunaires sur la croissance des cultures sont rares, et leurs résultats sont souvent mitigés. Les facteurs environnementaux tels que la température, l'humidité, la lumière et les nutriments ont un impact bien plus significatif sur la croissance des plantes que les phases lunaires.

Pratiques adaptées à l'environnement et aux cultures : Les agriculteurs modernes se

fient principalement à des pratiques agricoles basées sur des données scientifiques et sur les besoins spécifiques de chaque culture. Ils ajustent leurs pratiques en fonction du climat, du sol et des besoins des cultures plutôt que de suivre des calendriers basés sur les phases lunaires.

1.6 Conclusion

En résumé, bien que le lien entre la culture des potagers et le cycle lunaire ait été un aspect traditionnel de l'agriculture, la science moderne et les pratiques agricoles basées sur des données ont pris le pas. La majorité des agriculteurs se tournent vers des méthodes plus scientifiques pour maximiser les rendements et la qualité des cultures, tout en prenant en compte les caractéristiques locales et environnementales. Cependant, les croyances liées au cycle lunaire restent ancrées dans certaines cultures et sont parfois utilisées par des jardiniers amateurs ou des agriculteurs traditionnels.

Dans un jardin, il est essentiel d'adopter une approche pragmatique et réaliste de la culture des plantes. Plutôt que de s'en tenir strictement à des théories ou des modèles préétablis, il est important d'observer la nature telle qu'elle est. Les théories peuvent être utiles comme lignes directrices, mais elles ne doivent pas être la seule source de guidance.

En résumé, dans un jardin, il est important de se baser sur l'observation et l'expérience plutôt que de simplement suivre des théories. Les jardiniers doivent faire des choix en fonction de leur environnement spécifique et de leurs objectifs, tout en encourageant une compréhension profonde et une appréciation de la nature. Les jardins peuvent ainsi devenir des espaces de découverte constante et de connexion avec le monde naturel qui nous entoure.

Chapitre 2

Android

Sommaire

2.1 Description	25
2.1.1 Principale contrainte	25
2.1.2 Couche d'android	27
2.2 Le langage Java	28
2.2.1 Les variables	28
2.2.2 Compilation	29
2.3 Composition de fichier	30
2.3.1 Gestion de ressource	31
2.3.2 Les quantificateur	32
2.4 Environnement de développement	33
2.4.1 Générer de paquet	33
2.4.2 Classe R	33
2.4.3 Les composants	34
2.5 Pattern d'Android	35
2.6 Conclusion	35

2.1 Description

Android est un système d'exploitation open source développé principalement par Google pour une variété de dispositifs, principalement les smartphones et les tablettes. Voici quelques points clés pour comprendre Android :

- **Système d'exploitation mobile :** Android est un système d'exploitation conçu spécifiquement pour les appareils mobiles. Il offre un environnement informatique complet, incluant le système d'exploitation, le middleware et les applications de base. Android est basé sur le noyau Linux, ce qui lui confère une stabilité et une sécurité accrues.
- **Open Source :** Android est un projet open source, ce qui signifie que son code source est disponible et peut être modifié par les développeurs. Cela favorise la flexibilité, l'innovation et la collaboration.
- **Écosystème d'applications :** L'une des caractéristiques distinctives d'Android est son vaste écosystème d'applications. Les développeurs peuvent créer des applications pour Android en utilisant le langage de programmation Java (ou Kotlin) et les distribuer via la plateforme Google Play Store.



FIGURE 2.1 – Logo Android

2.1.1 Principale contrainte

Le développement d'applications sous Android offre de nombreuses opportunités, mais il comporte également des défis et des contraintes. Voici quelques-unes des contraintes courantes associées au développement sous Android :

- **Fragmentation :** Comme mentionné précédemment, l'écosystème Android est fragmenté en raison de la diversité des fabricants, des modèles et des versions du système d'exploitation en circulation. Cela signifie que les développeurs doivent prendre en compte cette diversité lors de la conception et du test de leurs applications pour assurer une compatibilité optimale.
- **Compatibilité des Appareils :** En raison de la diversité matérielle, certaines applications peuvent ne pas fonctionner de manière optimale sur tous les appareils Android en raison des variations de taille d'écran, de résolution, de puissance de traitement, etc.
- **Mises à Jour du Système d'Exploitation :** Les mises à jour d'Android ne sont pas toujours déployées uniformément sur tous les appareils, car cela dépend des fabricants et des opérateurs. Certains utilisateurs peuvent continuer à utiliser des versions plus anciennes d'Android, ce qui peut rendre difficile la prise en charge de certaines fonctionnalités récentes dans les applications.
- **Sécurité :** En raison de la nature open source d'Android, il peut être plus vulnérable aux attaques de sécurité que des systèmes plus fermés. Les développeurs doivent prendre des mesures supplémentaires pour sécuriser leurs applications et protéger les données des utilisateurs.
- **Optimisation des Performances :** En raison de la diversité des dispositifs, il peut être plus complexe d'optimiser les performances des applications pour garantir une expérience utilisateur fluide sur toutes les plateformes.
- **Distribution des Applications :** Bien que la distribution via Google Play soit la méthode la plus courante, certains marchés et régions peuvent avoir des restrictions ou des préférences spécifiques en matière de distribution, ce qui peut poser des défis pour la portée globale de l'application.
- **Ressources Limitées :** Certains appareils Android peuvent avoir des ressources limitées en termes de mémoire, de puissance de traitement et de stockage. Les développeurs doivent optimiser leurs applications pour fonctionner efficacement sur une gamme variée de matériel.

2.1.2 Couche d'android

Le développement Android repose sur un ensemble de couches logicielles qui fournissent les fonctionnalités nécessaires pour créer des applications. Ces couches constituent l'architecture du système d'exploitation Android. Voici les principales couches d'Android, de la plus basse à la plus haute :

- **Noyau Linux** : Android est construit sur le noyau Linux, qui fournit les fonctionnalités de base du système d'exploitation, telles que la gestion de la mémoire, la gestion des processus, la gestion du système de fichiers, et la communication avec le matériel.
- **Bibliothèques Android** : Cette couche comprend un ensemble de bibliothèques écrites en langage C/C++ qui sont utilisées par divers composants du système Android. Ces bibliothèques fournissent des fonctionnalités telles que la gestion des graphiques, l'accès à la base de données SQLite, la manipulation d'images, etc.
- **Android Runtime (ART)** : Il s'agit de l'environnement d'exécution des applications Android. Il utilise le bytecode DEX (Dalvik Executable) pour exécuter les applications. À partir d'Android 5.0, ART a remplacé Dalvik comme moteur d'exécution par défaut.
- **Framework d'Application** : Cette couche fournit les classes et les services nécessaires au développement des applications Android. Elle inclut des composants tels que les activités, les services, les fournisseurs de contenu, les gestionnaires d'applications, etc. Les développeurs utilisent ce framework pour construire l'interface utilisateur et la logique métier de leurs applications.
- **Applications Système** : Au sommet de la pile se trouvent les applications système préinstallées sur chaque appareil Android. Cela inclut des applications telles que le lanceur (interface utilisateur), le gestionnaire d'applications, le navigateur, le gestionnaire de contacts, etc. Ces applications offrent une expérience utilisateur de base et servent de référence pour les développeurs d'applications tierces.

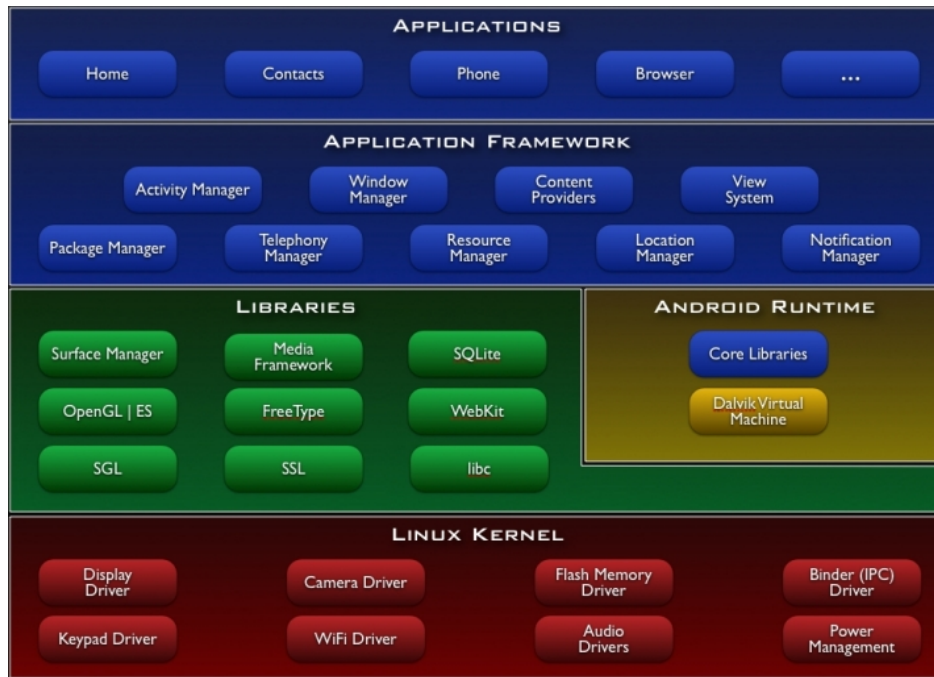


FIGURE 2.2 – Couche android

2.2 Le langage Java

Java est un langage de programmation de haut niveau, orienté objet et indépendant de la plateforme. Il a été créé par Sun Microsystems (maintenant une division d'Oracle) et a été rendu public en 1995.

2.2.1 Les variables

En informatique, les variables sont des symboles qui associent un nom à une valeur. Il existe deux types que l'on utilisera :

Type Primitive : Ce sont des types de donnée non objet qu'on retrouve dans de nombreux autres langages de programmation. A chacun de ces types correspond un type objet appelé «type enveloppe».

Ex : float,int

Objet : L'objet est comparable au tableau à la seule différence que chaque variable stockée n'est pas appelée par un index mais par un nom.

En Java, la déclaration d'une classe, d'une méthode ou d'un membre peut être précédée par un modificateur d'accès ou attribut.

Un modificateur indique si les autres classes de l'application pourront accéder ou non à la classe/méthode/membre Une classe a une visibilité :

- **public :** sa définition est précédée de public, et il peut être utilisé par tout utilisateur

de la classe.

- **privé** : sa définition est précédée de `private`, et il ne peut être utilisé qu'à l'intérieur de la classe
- **protégé** : sa définition est précédée de `protected`, et il ne peut être utilisé qu'à l'intérieur de la classe, ou des classes dérivées.
- **paquetage** : l'attribut peut être utilisé dans toute classe du même paquetage.

par défaut : sans modificateur d'accès, seules les classes du même package peuvent accéder à l'item.

Il y a aussi d'autre attribut pour les variable type objet comme :

- **Abstract** : une classe abstraite est une classe dont l'implémentation n'est pas complète et qui n'est pas instanciable. Elle sert de base à d'autres classes dérivées (héritées).
- **final** : indique qu'un élément ne peut être changé dans la suite du programme.
- **Static** : Une méthode statique est une méthode qui peut être appelée même sans avoir instancié la classe.

2.2.2 Compilation

Un compilateur Java est un compilateur pour le langage de programmation Java. Le format de sortie le plus courant pour un compilateur Java est des fichiers `.class` contenant le bytecode Java plate-forme agnostique. Il existe aussi des compilateurs produisant du code machine optimisé pour une combinaison matériel/système d'exploitation particulière.

La machine virtuelle Java (JVM) charge les fichiers `.class` et interprètes le bytecode ou le compile à la volée et peut également l'optimiser en utilisant la compilation dynamique. Ainsi on utilise un Java Development Kit (JDK) ou on peut trouver JRE (Java Runtime Environment) contient JVM (Java virtual Machine) .

Exploiter une nouvelle plate-forme n'est jamais chose aisée. C'est pourquoi Google fournit, en plus du système d'exploitation, un kit de développement .Ce SDK est un ensemble d'outils logiciels qui permet aux développeurs et aux entreprises de créer des applications.

Un kit de développement logiciel, aussi appelé trousse de développement logiciel, facilite donc le développement d'un logiciel sur une plateforme donnée.

Et aussi d'un API qui est un ensemble de classes regroupant des fonctions mises à disposition des développeurs. Ces fonctions ou méthodes peuvent être regroupées dans des bibliothèques logicielles ou des services. Le plus souvent, elles effectuent des traitements de bas niveau et proposent au développeur une interface de plus haut niveau pour qu'il puisse accéder à des fonctionnalités plus facilement et surtout plus immédiatement.

La forme de paquet SDK est : **Android[nombre](API[un autre nombre])**

Cette SDK est aussi suivi d'un ADT(Android Development Tools) qui ajoute le support intégré pour Android projets et des outils. Le plugin ADT comprend une variété d'extensions puissantes qui font création, l'exécution et le débogage Android des applications plus rapidement et plus facilement.

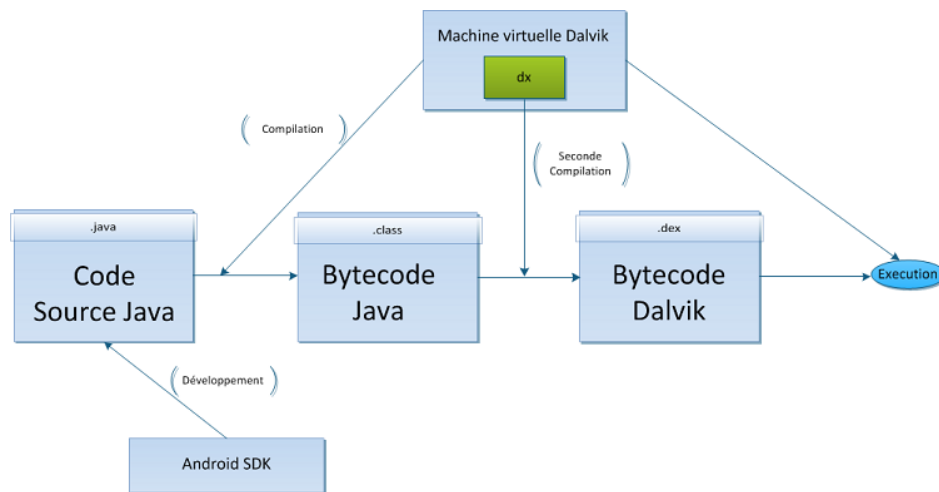


FIGURE 2.3 – Compilation

2.3 Composition de fichier

Donc un projet et généralement composer de :

R.java : Fichier contenant les "paramètres" de l'application. Il se génère automatiquement.

Bin : Le dossier "Bin" contient les binaires de l'application

Manifest : description de l'application

src : les fichiers source, une fois compiler R.java est créé qui fournira des infos res

libs : bibliothèque

res : Ce dossier contient les ressource, les images utilisées dans le projet, les XML du projet (les fichiers XML servent à définir l'interface de l'application) et le dossier "values".

Généralement dans les ressources il y a :

- ressource :xml
- Drawable
- Layout
- Menu
- Mipmap
- Value
- image accessible par R

gen : fichier par ADT(Android development tools)

assets/ : ressource brutes (rawbytes) accessible par flux de donner

andriid Manifest.xml : décrivent l'application, fichier XML dans lequel on déclare les différentes activités de l'application, et certains paramètres de l'application (par exemple si on autorise la rotation de l'application, le sens de l'application (vertical, horizontal), l'icône de l'application, etc..).

default.propretie : propriété pour la compilation

2.3.1 Gestion de ressource

Langage de balisage est un langage qui s'écrit grâce à des balises. Ces balises permettent de structurer de manière hiérarchisée et organisée les données d'un document. Ici l'application utilise le xml. **res/** : stockage des ressource pour s'adapté au divers changement

```
1 //debut de fichier
2 <?xml version="1.0" encoding="utf-8" ?>
```

Il y a des nœud parents et enfant **res/drawable** : Ces dossiers contiennent les images ou les images matricielles de l'application. Tous ces dossiers devraient, dans l'idéal, contenir les mêmes images, mais avec des résolutions différentes. En effet, selon la résolution de l'écran utilisé, les images seront tirées du dossier correspondant.

res/value : ce dossier contient aussi des XML, mais qui servent à stocker des chaînes de caractères.

res/layout : Ce dossier contient les XML de l'application pour les interfaces. Vous pouvez déjà voir le main.xml, qui correspond à l'interface de l'activité principale de l'application.

res/menu : XML, menu

res/raw : données brute

res/value : catégorie nombreuse

Ex : variable

la forme générale est donc :

```
1 < ?xml version="1.0" encoding="utf-8"?>
2 <racine>
3     <!--commentaire-->
4     <elements      attribut1="valeur_1"
5 attribut2="valeur_2">
6 <feuille1 attribut3="valeur_3"/>
7 <feuille2> texte </feuille2>
8     </element>
9 </racine>
```

2.3.2 Les quantificateur

Les quantificateurs sont utilisés pour cibler précisément un certain nombre de priorités; à savoir la langue et la région, la taille de l'écran, l'orientation de l'écran, la résolution de l'écran et la version d'Android Les restrictions sont représentées par des quantificateurs et ce sont ces quantificateurs qui vous permettront de préciser le matériel pour lequel les fichiers dans ce répertoire sont destinés. La syntaxe à respecter peut être représentée ainsi :

```
1 res/<type_de_ressource>[<-quantificateur 1><-quantificateur 2>...<-quantificateur N>]
```

langue et région priorité 2 ex :fr

taille de l'écran :priorité :3 small,nomal,large,xlarge

orientation de l'écran priorité :5 port,land

résolution de l'écran priorité 8 :

- ldpi, 120dpi
- mdi 160 dpi
- hdmi 240dpi
- xhdpi 320dpi
- nodpi non redimensionee

Version d'Android priorité 14 ordre des priorité a respect croissante

2.4 Environnement de développement

Gradle script : outil de compilation

logCat : message détailler debugge, info, erreur...

```
1      Log.i(String tag, String message)
2      Log.w(tag, message)
3      Log.e(tag, message)
```

ADB : Android Debugge Bridge passerelle PC et devise gestion serveur :

-adb start-server : démarre

-adb kill-server : arrêt

-adb device : liste des connectées

2.4.1 Générer de paquet

Build>generate Signed APK

Clé privé

Pour avoir une clé privée il faut un key store

Cree un key store :

Remplir

Cree clé privée

Emplacement du fichier

2.4.2 Classe R

Classe R La classe R se trouve :

App>build>generated>source>r>debug

Drawable :

String :

Pour récupérer les sources

Private String variable=null;

Variable =getResource().getString(R.String.hello); **Norme de nom**

Android : nom attribut

Il est possible de créer une interface par codage mais vaut mieux le stocker dans res/-

layout/main.xml, mettre chaque type dans le dossier qui le convient

Ex :string res/value/strings.xml

@id/nom : utilise R.id.nom référence définit ailleurs

@+id/nom crée cet identifiant

2.4.3 Les composants

Vue : élément de l'interface graphique

Contrôles : sous-ensemble de vue

Activité : écran structure

ex : des vue a l'intérieur d'un vue

Xml : fichier de configuration

forme générale de fichier class d'une application

```
1 Package projet.memoire.package ;
2 Import android.os.bundle ;
3 Import android.view.menu ;
4 Import android.view.MenuItem ;
5 Import android.support.v4.app.NavUtils ;
6 Public class MainActivity extends Activity{
7     @Override //facultative
8     Public void onCreate(Bundle savedInstanceState){
9         Super.onCreate(savedInstanceState) ;
10 setContentView(R.layout.activity_main) ;
11 }
12 @Override//indique la redefinition de methode
13 Public boolean onCreateOptionsMenu(Menu menu) {
14     getMenuInflater().inflate(R.menu.activity_main,menu) ;
15 return true ;
16 }
17 }
```

Les ressource Nous pouvons distinguer plusieurs types de ressources :

- les fichiers de ressources de l'application (images, chaînes de caractères, layout, xml) qui dépendent du contexte (langue française, taille de l'écran, mode portrait/-paysage...);
- les bases de données;
- les fichiers préférences;
- les fichiers pour la lecture et la lecture/écriture sans contexte.

2.5 Pattern d'Android

MVC (Modèle-Vue-Contrôleur) : Dans le modèle MVC, le code est organisé en trois composants principaux :

- **Modèle** : Représente la logique métier et les données de l'application.
- **Vue** : Affiche l'interface utilisateur et interagit avec l'utilisateur.
- **Contrôleur** : Gère les interactions de l'utilisateur, manipule les données du modèle et met à jour la vue.

strucbasique Dans le contexte Android, l'activité peut être considérée comme le contrôleur, la vue est l'interface utilisateur (XML) et le modèle est généralement représenté par des classes qui gèrent les données de l'application.

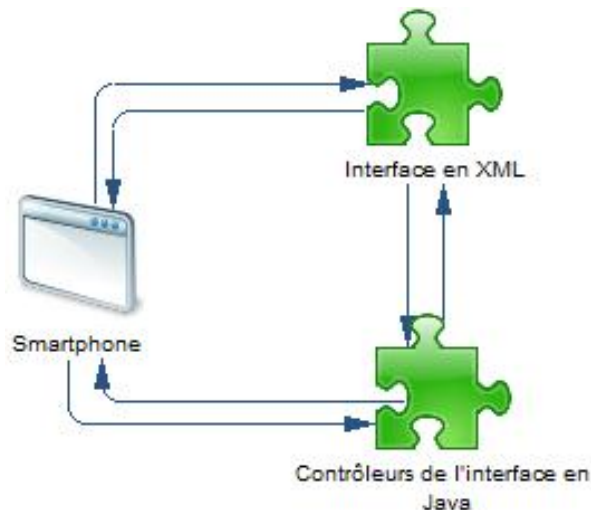


FIGURE 2.4 – MVC Android

2.6 Conclusion

Pour développer sur Android il faut plusieurs éléments, pour combler ces éléments on utilise le SDK approprié pour faire apparaître les éléments voulus par la version voulue du point de vue générale, on a besoin tout simplement des ressources, de la classe R et d'un Modèle. Et on constate que le développement d'Android est totalement dans le monde de MVC.

Chapitre 3

Implantation du programme

Sommaire

3.1 Cycle de vie d'une activité	37
3.2 Les intentes	39
3.3 Modelisation du programme	39
3.3.1 Organigramme pour agriculture biodynamique	39
3.3.2 Diagramme de cas d'utilisation	41
3.3.3 Diagramme de classes	43
3.3.4 Diagramme de séquence	45
3.3.5 Diagramme d'activités	47
3.4 Présentation de l'application	50
3.5 Conclusion	53

Une application Android est un assemblage de composants liés grâce à un fichier de configuration. Activité : fenêtre dont il est possible de naviguer, il contient les contextes (lien entre système et autres) pour le gérer on a une liste LIFO :

Les Activités est extends activity

Les vue est extend view

3.1 Cycle de vie d'une activité

En Android, une activité est une composante de l'application qui représente une seule fenêtre, avec une interface utilisateur, sur l'écran de l'appareil. Chaque activité a un cycle de vie qui décrit son état à différents moments, depuis sa création jusqu'à sa destruction. Le cycle de vie d'une activité est géré par le système Android et suit un ensemble de méthodes spécifiques qui sont appelées à différentes étapes. Voici les principales étapes du cycle de vie d'une activité :

- **Création (onCreate())** : Cette méthode est appelée lorsque l'activité est d'abord créée. C'est généralement l'endroit où vous effectuez l'initialisation de base, telle que l'inflation de la mise en page (layout) et l'initialisation des objets nécessaires.
- **Démarrage (onStart())** : Après la création, l'activité passe à l'état de démarrage. À ce stade, l'activité devient visible à l'utilisateur, mais elle n'est pas encore interactive.
- **Reprise (onResume())** : L'activité est maintenant en état de reprise. C'est le moment où elle devient active et l'utilisateur peut interagir avec elle. Cette méthode est également le meilleur endroit pour enregistrer des gestionnaires d'événements ou des mises à jour de l'interface utilisateur.
- **Mise en arrière-plan (onPause())** : L'activité est en cours d'entrée en arrière-plan. Elle reste visible, mais elle n'accepte généralement plus d'entrées de l'utilisateur. À ce stade, vous pouvez enregistrer les données nécessaires pour restaurer l'état de l'activité si nécessaire.
- **Arrêt (onStop())** : L'activité n'est plus visible à l'utilisateur. À ce stade, elle est en cours d'arrêt et peut être arrêtée par le système pour libérer des ressources.
- **Redémarrage (onRestart())** : L'activité est en cours de redémarrage après avoir été arrêtée. Elle est sur le point de revenir à l'état de reprise.

- **Démarrage (onStart()) et Reprise (onResume())** : Si l'activité passe par un redémarrage, elle retourne ensuite à l'état de démarrage et de reprise.
- **Arrêt (onStop())** : Si l'activité n'est plus visible à l'utilisateur (par exemple, si une autre activité a été lancée), elle passe à l'état d'arrêt.
- **Destruction (onDestroy())** : L'activité est détruite. Cela se produit lorsque l'activité est explicitement fermée par l'utilisateur ou lorsque le système décide de libérer des ressources. C'est le dernier point du cycle de vie de l'activité.

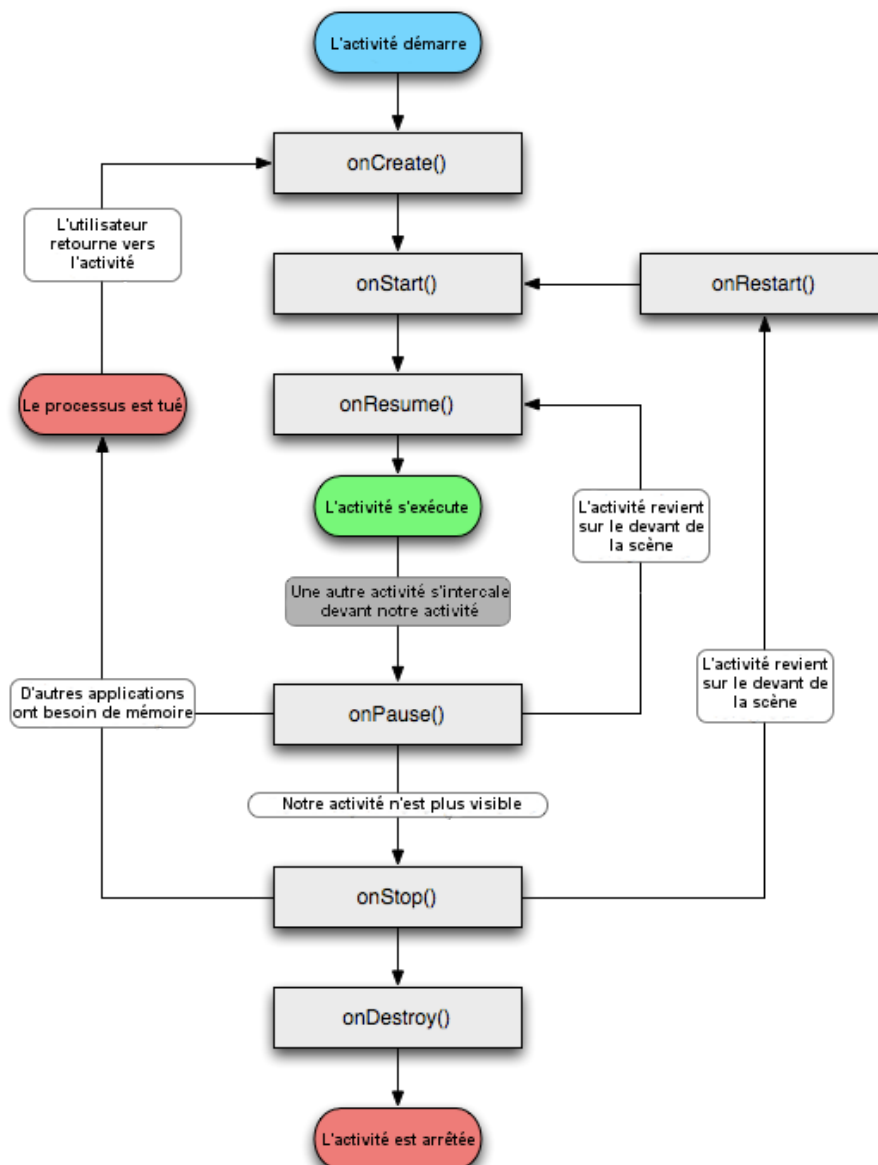


FIGURE 3.1 – Cycle de vie d'une activité

3.2 Les intents

Pour communiquer c'est-à-dire faire circuler des messages d'une application à une autre ou à l'intérieur d'une même application il faut savoir plusieurs choses. Il y a des agents qui sont chargés de ce mécanisme d'échange s'appellent les intents. Ce mécanisme est tellement important qu'Android lui-même l'utilise massivement en interne.

Un intent est en fait un objet qui contient plusieurs champs, représentés à la figure suivante

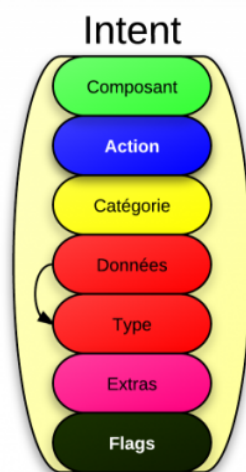


FIGURE 3.2 – Une intent

3.3 Modelisation du programme

3.3.1 Organigramme pour agriculture biodynamique

L'agriculture biodynamique est bien plus qu'une simple méthode de culture ; c'est un système holistique qui intègre des pratiques agricoles traditionnelles avec une compréhension profonde des cycles naturels, y compris le cycle lunaire. Voici comment ce système fonctionne en harmonie avec les phases de la lune pour produire des cultures saines et durables. L'objectif principal de cet organigramme est de faciliter la décision quant aux plantes à planter à des dates spécifiques, en se basant sur la phase lunaire correspondante. Le processus commence par la saisie d'une date précise, suivie de l'identification de la phase lunaire associée à cette date. Ensuite, en fonction de cette phase lunaire, l'organigramme détermine le type de plante le plus approprié à planter : feuille, fleur, racine ou bourgeon. Cette approche permet une gestion plus efficace des cultures en tenant

compte des influences lunaires sur la croissance des plantes, contribuant ainsi à optimiser les rendements agricoles tout en respectant les cycles naturels.

- **Initialisation des paramètres :**

Avant de commencer toute activité agricole, les agriculteurs biodynamiques prennent en compte le calendrier lunaire pour déterminer les différentes phases de la lune, y compris la nouvelle lune, le premier quartier, la pleine lune et le dernier quartier.

- **Planification des activités agricoles :**

En se basant sur les principes de la biodynamique, un calendrier des activités agricoles est établi, en tenant compte des phases de la lune. Pendant la période de la nouvelle lune jusqu'au premier quartier, les activités favorisent la croissance des parties aériennes des plantes. Entre le premier quartier et la pleine lune, les activités sont axées sur la floraison et la fructification. Pendant la période entre la pleine lune et le dernier quartier, l'accent est mis sur la croissance des racines et la consolidation des fruits. Enfin, entre le dernier quartier et la nouvelle lune, les activités se concentrent sur la préparation du sol, la gestion des mauvaises herbes et la récolte des cultures à maturité.

- **Application des préparations biodynamiques :**

Pendant les périodes propices du cycle lunaire, des préparations biodynamiques spécifiques sont appliquées au sol et aux cultures, en tenant compte des recommandations spécifiques pour chaque phase de la lune. Ces préparations sont conçues pour dynamiser le sol et favoriser la santé des plantes.

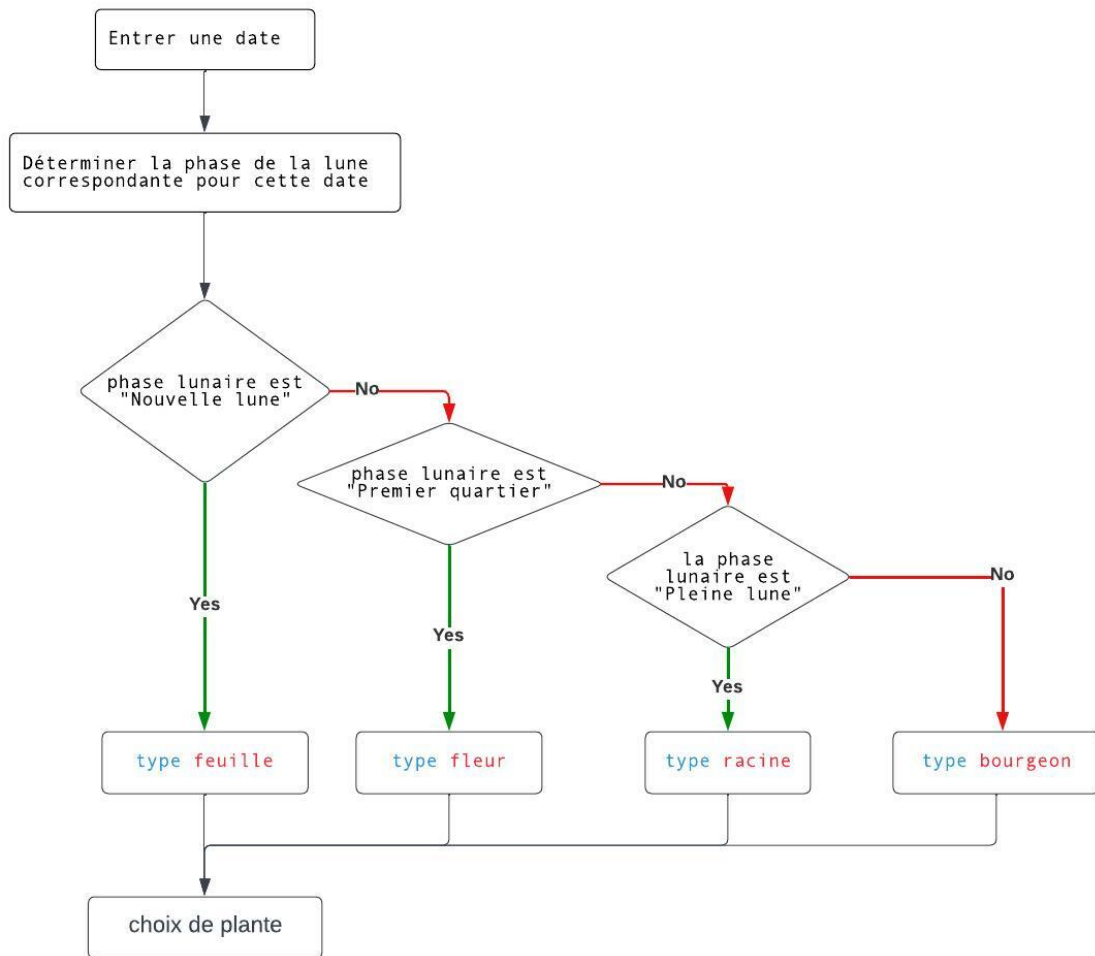


FIGURE 3.3 – Organigramme de calcul de phase de la lune

3.3.2 Diagramme de cas d'utilisation

Il permet la visualisation des interactions entre objets du système La structure d'une opération est découpé en actions. C'est un des moyens de représenter les besoins à couvrir.

Le système de gestion de plantes potagères offre une plateforme complète pour répondre aux besoins variés des utilisateurs impliqués dans la culture et la gestion des plantes. Les jardiniers, qui sont responsables de l'entretien quotidien et de la croissance des plantes, peuvent utiliser le système pour planifier et suivre les tâches telles que la plantation, l'arrosage, la fertilisation et la récolte des légumes.

Les gestionnaires, en charge de superviser l'ensemble du processus de gestion des plantes, ont la possibilité de gérer les utilisateurs en ajoutant, modifiant ou supprimant des comptes selon les besoins de l'équipe de jardinage. Ils peuvent également gérer les

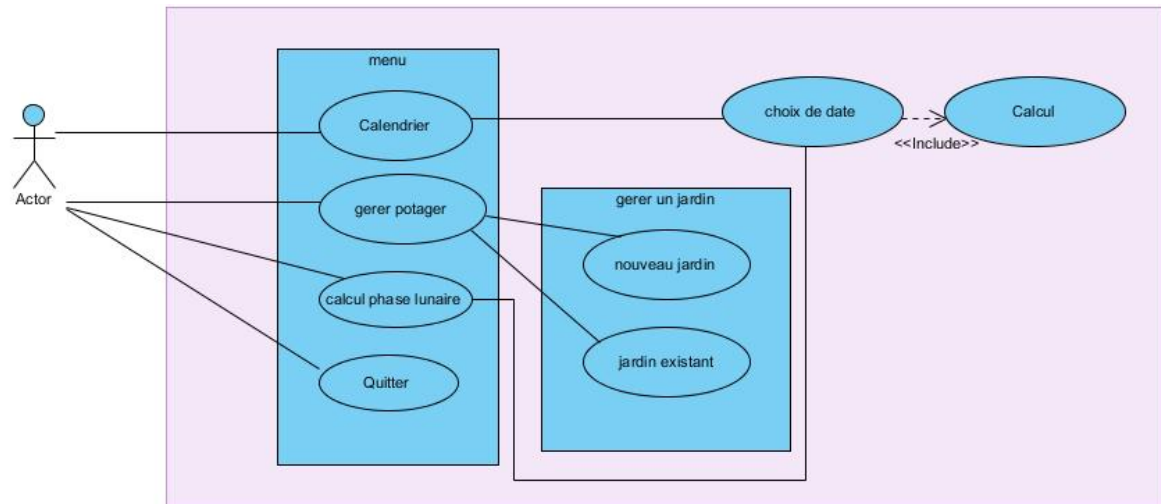


FIGURE 3.4 – Diagramme de cas d'utilisation

plantes en ajoutant de nouvelles espèces au catalogue, en modifiant les détails des plantes existantes et en supprimant celles qui ne sont plus cultivées.

Les experts en botanique, qui fournissent des conseils spécialisés sur les soins des plantes, peuvent utiliser le système pour consulter les informations détaillées sur chaque espèce de plante, y compris leurs besoins en lumière, en eau et en nutriments, ainsi que les meilleures pratiques de culture.

Ces activités sont soutenues par des inclusions telles que l'affichage d'informations pertinentes sur les spécifications des plantes, le calcul précis des quantités d'eau et de nutriments nécessaires pour chaque plante, et la possibilité d'effectuer des opérations CRUD complètes pour gérer les données de manière efficace.

Le système démarre dès qu'un utilisateur se connecte, offrant la possibilité de sélectionner une date spécifique pour planifier des tâches de jardinage via un calendrier interactif intégré. Une fois dans le menu Calendrier, l'utilisateur peut choisir une date et lancer un calcul pour déterminer les différentes phases lunaires associées à cette date. Il peut également créer et gérer un potager, en ajoutant de nouvelles plantes, en modifiant les détails des plantes existantes et en supprimant celles qui ne sont plus cultivées.

Enfin, l'option de quitter le système est disponible pour permettre aux utilisateurs de terminer leurs tâches et de quitter l'interface en toute simplicité une fois qu'ils ont accompli leurs actions.

3.3.3 Diagramme de classes

Le diagramme de classe est une représentation visuelle de la structure des données du système, organisée en classes et en relations entre ces classes. Nous avons mis en place plusieurs classes dans notre programme afin de suivre le modèle MVC (Modèle-Vue-Contrôleur). Ce modèle permet de séparer les différentes responsabilités de l'application : le modèle pour la manipulation des données, la vue pour l'affichage des informations à l'utilisateur, et le contrôleur pour la gestion des interactions et la coordination entre le modèle et la vue. Cette approche de conception nous permet de mieux organiser notre code et de faciliter la maintenance et l'évolution du système.

- classe principale :

La classe principale de notre application est la classe Main. Elle constitue le point d'entrée initial de notre programme et représente la première interaction entre l'utilisateur et l'application. Cette classe gère l'initialisation de l'application et coordonne les différentes fonctionnalités en interagissant avec l'utilisateur via l'interface utilisateur ou d'autres systèmes. En tant que point de départ de l'application, la classe Main est essentielle pour orchestrer le flux de contrôle et faciliter la communication entre l'utilisateur et le système, assurant ainsi une expérience utilisateur fluide et cohérente.

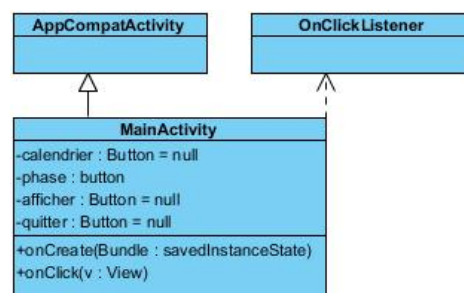


FIGURE 3.5 – Diagramme de classes main

- La classe Controller base de données :

La classe Controller Base De Donnees représente le contrôleur dans notre application. Elle agit comme un intermédiaire entre la vue et le modèle, facilitant la communication entre l'utilisateur et la base de données. Cette classe gère les requêtes

et les actions de l'utilisateur, puis transmet ces informations au modèle pour effectuer les opérations correspondantes sur la base de données. Le modèle, quant à lui, est une classe de gestion de base de données qui permet d'effectuer des opérations telles que l'ajout, la mise à jour et la suppression de données dans la base de données. Il agit comme une interface entre l'application et la base de données, garantissant une manipulation efficace et sécurisée des données.

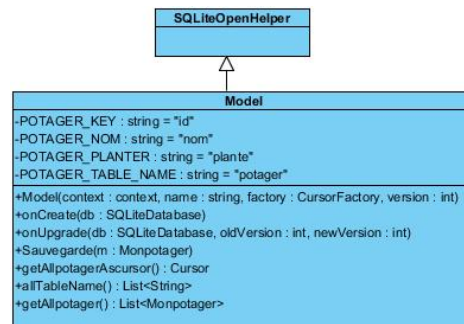


FIGURE 3.6 – Diagramme de classes Model

- La classe type View :

La classe type View représente la vue dans notre application. Le diagramme de classe de la liste de potager sert à afficher les listes de potagers que l'on peut cultiver avec une date sélectionnée. À l'aide de l'objet `ListView`, cette classe gère précisément une liste. Une fois qu'un élément est sélectionné, il est enregistré interface entre l'application et la base de données, garantissant une manipulation efficace et sécurisée des données.

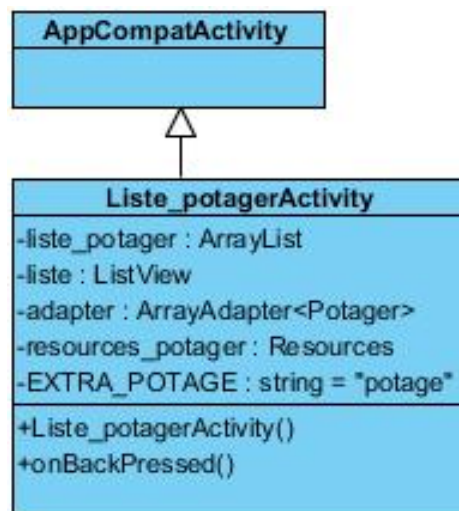


FIGURE 3.7 – Diagramme de classe de liste de potager

3.3.4 Diagramme de séquence

Le diagramme de séquence, outil crucial en conception logicielle, offre une représentation visuelle des interactions temporelles entre les différents objets impliqués dans la réalisation d'une interface homme-machine. Il joue un rôle essentiel en permettant de décrire de manière détaillée les scénarios d'utilisation, chaque cas d'utilisation étant accompagné d'un scénario décrivant chronologiquement les actions d'un acteur sur le système.

Ce diagramme offre une compréhension approfondie à la fois sur la nature des informations échangées entre les objets et sur l'ordonnancement précis des événements qui se produisent lors de l'exécution du scénario.

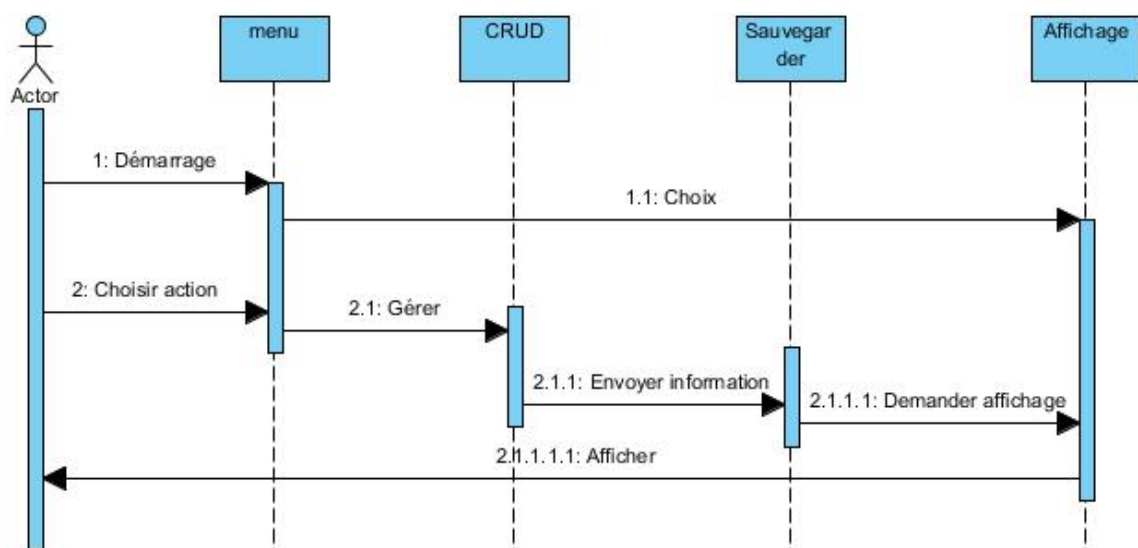


FIGURE 3.8 – Diagramme de séquence générale

Dans le contexte de la séquence de gestion d'un nouveau jardin implique l'interaction de l'utilisateur avec l'interface logicielle pour créer et gérer un nouvel espace de jardinage. Cette séquence comprend plusieurs étapes pour faciliter la création et la gestion efficace du jardin : Tout d'abord, l'utilisateur lance l'application de gestion de jardin potager et sélectionne l'option pour créer un nouveau jardin. Ensuite, une fois que les détails du jardin ont été enregistrés, l'utilisateur est dirigé vers la fonctionnalité qui lui permet d'explorer une liste de plantes disponibles pour la culture dans son jardin.

L'utilisateur peut ensuite parcourir cette liste de plantes et sélectionner.

Une fois que l'utilisateur a choisi les plantes à cultiver, l'interface lui permet de les ajouter à son jardin virtuel.

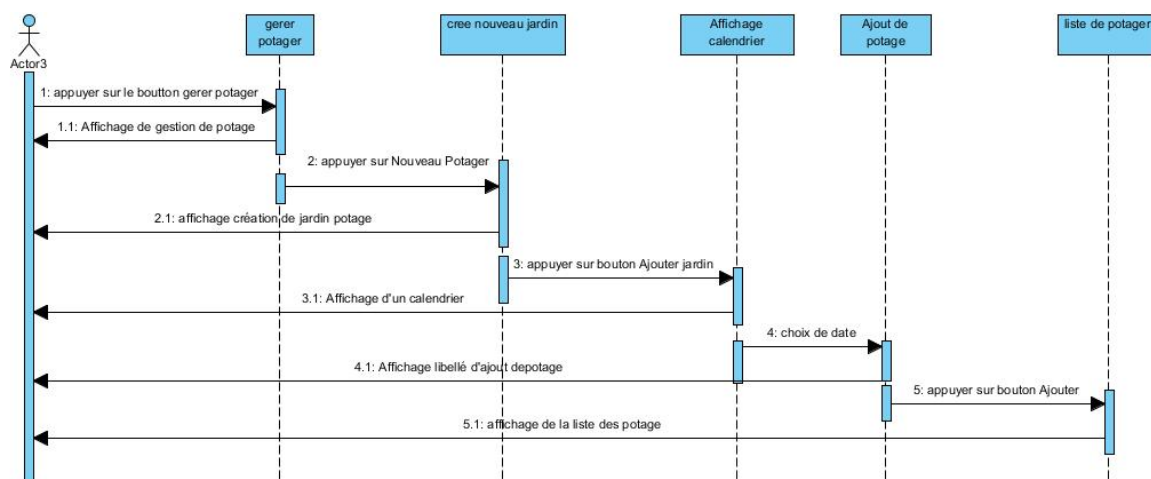


FIGURE 3.9 – Diagramme de séquence gestion de nouveau jardin

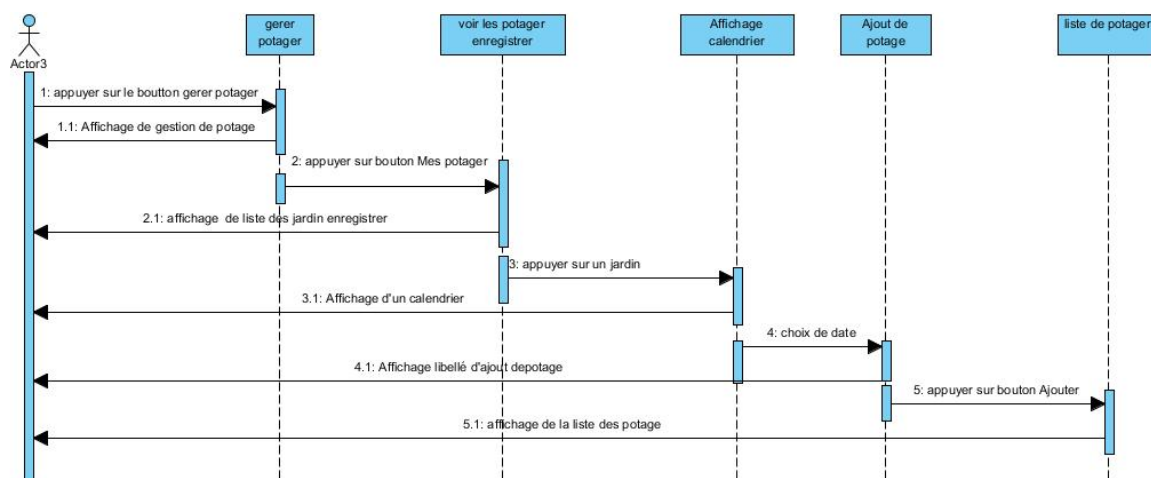


FIGURE 3.10 – Diagramme de séquence gestion de jardin enregistrer

Le diagramme de séquence relatif au calcul de la phase lunaire offre une représen-

tation détaillée des interactions entre les différents composants logiciels impliqués dans le processus de détermination de la phase actuelle de la lune. Ce processus commence généralement par l'acquisition de la date choisie par l'utilisateur. Ensuite, le logiciel effectue des calculs pour déterminer la position actuelle de la lune dans son orbite autour de la Terre, en utilisant des algorithmes astronomiques appropriés.

Une fois que la position de la lune est déterminée, le logiciel identifie la phase lunaire correspondante, comme la nouvelle lune, le premier quartier, la pleine lune ou le dernier quartier.

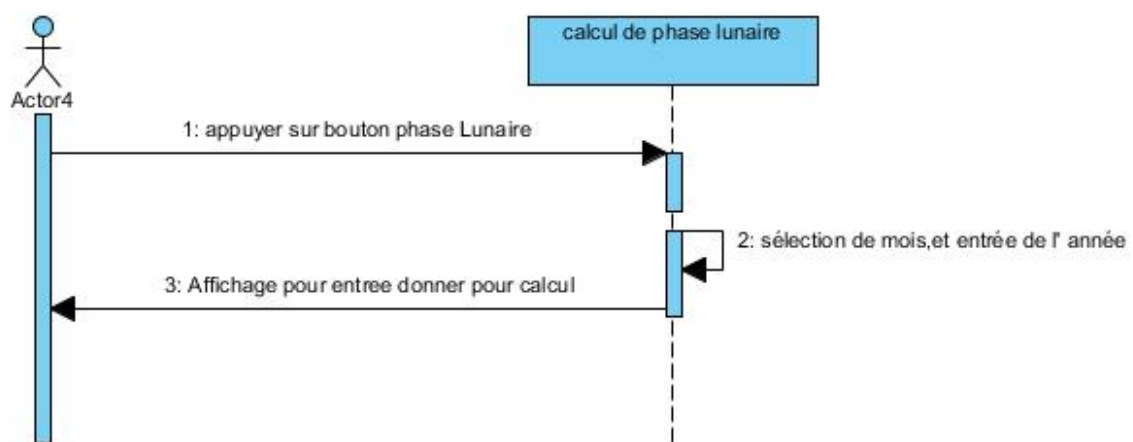


FIGURE 3.11 – Diagramme de séquence calcul du phase lunaire

3.3.5 Diagramme d'activités

Il permet la représentation du comportement entre les objets d'une même classe en termes d'états et de transitions d'états, qui est lié à celui du diagramme de séquence. Cette représentation permet de décrire comment un objet passe d'un état à un autre en réponse à des événements ou des actions. Contrairement au diagramme de séquence qui se concentre sur les interactions entre les objets à un moment donné, le diagramme d'états-transitions montre comment le comportement d'un objet peut changer au fil du temps, en fonction de son état interne et des stimuli externes, comme ici :

Le cycle de vie de l'activité de gestion d'un nouveau jardin englobe plusieurs étapes clés,

FIGURE 3.12 – Diagramme d'état générale

depuis sa création initiale jusqu'à la gestion quotidienne des données. Tout d'abord, la phase de création implique la mise en place initiale du jardin dans le logiciel de gestion,

où l'utilisateur peut saisir des informations telles que l'emplacement, les dimensions, et d'autres détails pertinents pour définir l'espace de jardinage. Ensuite, lors de la phase d'ajout de données, l'utilisateur peut enrichir le jardin en entrant des informations supplémentaires telles que les types de plantes prévues pour la culture, les dates de plantation, les méthodes de culture envisagées, et d'autres détails utiles. Cette étape permet de personnaliser davantage la planification et la gestion du jardin, en tenant compte des préférences et des besoins spécifiques de l'utilisateur. Ensuite, lors de la phase d'ajout de

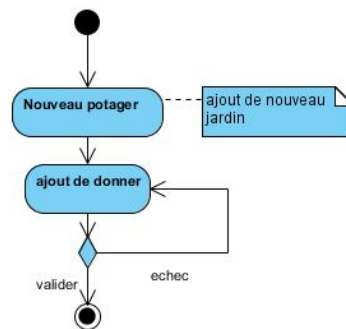


FIGURE 3.13 – Diagramme d'état gestion nouveau jardin

données, l'utilisateur peut enrichir le jardin en entrant des informations supplémentaires telles que les types de plantes prévues pour la culture, les dates de plantation, et d'autres détails utiles. Cette étape permet de personnaliser davantage la planification et la gestion du jardin, en tenant compte des préférences et des besoins spécifiques de l'utilisateur. Enfin, lors de la phase de gestion des données, l'utilisateur peut interagir avec le logiciel pour gérer et mettre à jour les informations du jardin au fur et à mesure que celui-ci évolue.

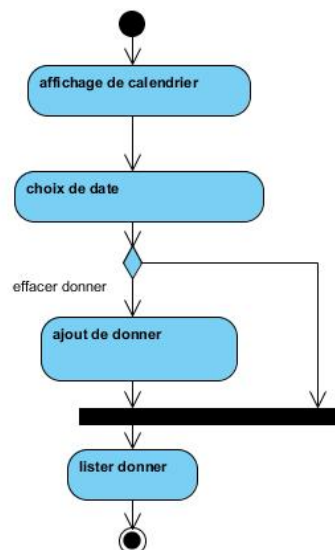


FIGURE 3.14 – Diagramme d'état tout le jardin

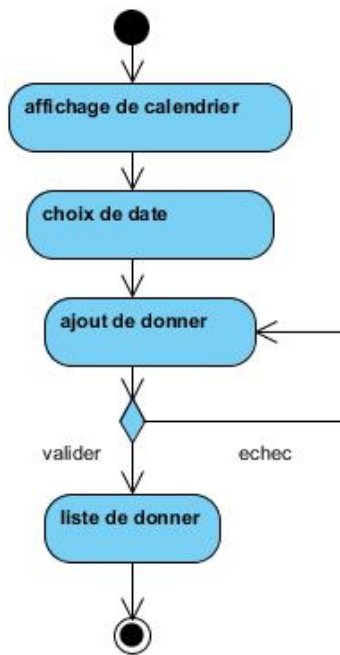


FIGURE 3.15 – Diagramme d'état du calendrier

Le diagramme d'état de la phase lunaire représente les différentes phases de la lune en fonction de la date entrée. Si la date est valide, le système calcule la phase lunaire correspondante et affiche les informations associées. En cas d'erreur, un message d'erreur est renvoyé pour indiquer que la date n'est pas valide.

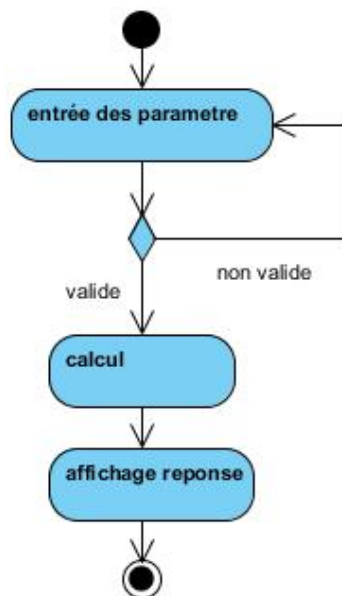


FIGURE 3.16 – Diagramme d'état de phase lunaire

3.4 Présentation de l'application

Une fois les étapes de modélisation et d'étude de toutes les séquences et l'organigramme de calcul fait, on passe par le développement de l'application. Une fois que toutes les étapes de modélisation, d'étude de toutes les séquences et acteurs, et d'organigramme de calcul ont été réalisées, on passe au développement de l'application. Cela implique de traduire les conceptions et les spécifications établies lors de la modélisation en code informatique fonctionnel. Pendant cette phase, à la création des fonctionnalités décrites dans les différents diagrammes et spécifications.

Voici le démarrage de l'application



FIGURE 3.17 – Splash screen

Lorsque l'application démarre, l'écran de démarrage (splash screen) s'affiche, offrant une première impression visuelle aux utilisateurs pendant le chargement initial de l'application. Une fois le chargement terminé, l'écran de démarrage laisse place au menu principal de l'application, offrant ainsi aux utilisateurs un point de départ pour explorer les fonctionnalités et les options disponibles. Le menu de l'application peut être conçu de manière à présenter de manière claire et intuitive les différentes sections ou fonctionnalités de l'application, permettant ainsi aux utilisateurs de naviguer facilement vers l'option de leur choix. En fournissant un écran de démarrage suivi d'un menu clair et convivial, l'application crée une expérience utilisateur fluide et engageante dès le début de l'utilisa-

tion.

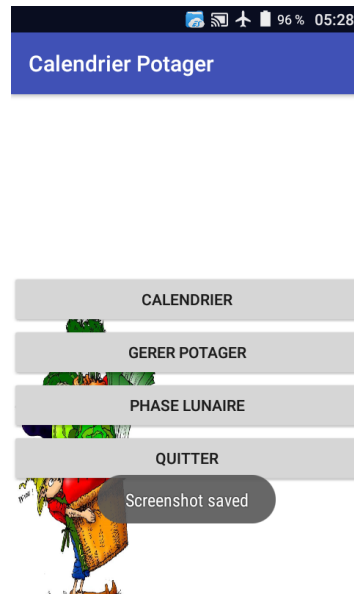


FIGURE 3.18 – Ecran d'accueil

Pour choisir une date, l'application propose un calendrier interactif où l'utilisateur peut sélectionner la date souhaitée. Cette fonctionnalité constitue la base de l'application, car elle permet à l'utilisateur de déterminer précisément le moment où il souhaite planifier ses activités de jardinage. Une fois la date sélectionnée, l'application affiche ensuite la liste des potagers adaptés à cette période spécifique. Cette liste est générée en fonction des recommandations de plantation appropriées pour cette période de l'année. En fournissant cette fonctionnalité de sélection de date et d'affichage de la liste des potagers, l'application aide les utilisateurs à planifier efficacement leurs activités de jardinage en fonction du calendrier saisonnier et des meilleures pratiques de culture.

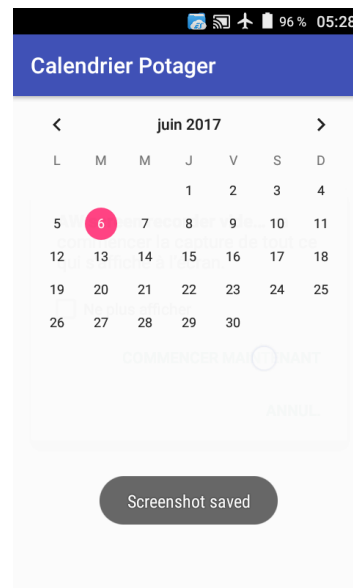


FIGURE 3.19 – Calendrier

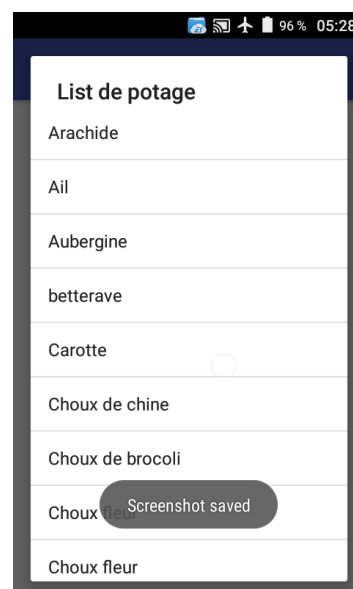


FIGURE 3.20 – Liste des potages

Ici, l'application affiche une liste des plantes disponibles pour la date sélectionnée. Une fois qu'un utilisateur a sauvegardé une liste de plantes pour une date spécifique, cette sauvegarde est conservée dans la base de données de l'application. L'utilisateur a alors la possibilité d'éditer cette liste ultérieurement. L'édition peut inclure des actions telles que le changement des plantes sélectionnées, leur suppression de la liste ou l'ajout de nouvelles plantes. Il est également important de noter que pour une date donnée, l'utilisateur peut sauvegarder plusieurs listes de plantes afin de pouvoir effectuer différentes simulations ou expérimentations dans son jardin. Cette fonctionnalité permet à l'utilisateur de planifier et de gérer efficacement ses activités de jardinage, en lui offrant la flexi-

bilité nécessaire pour ajuster ses plans en fonction de ses besoins et de ses préférences spécifiques.

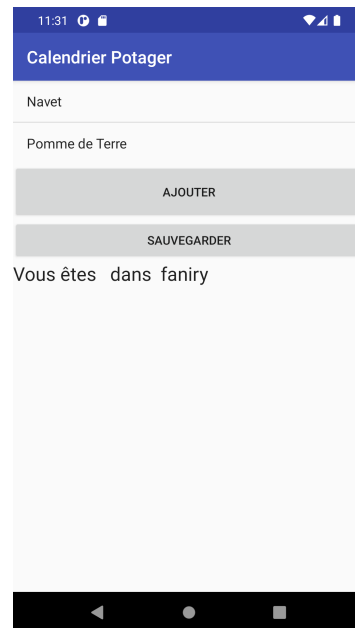


FIGURE 3.21 – Liste des potages

3.5 Conclusion

En conclusion, l'utilisation de plusieurs diagrammes pour modéliser l'application est essentielle pour une compréhension approfondie de son fonctionnement. Chaque diagramme, qu'il s'agisse d'un diagramme de séquence, d'un diagramme d'état ou d'un organigramme, offre une perspective unique sur les différents aspects de l'application. En examinant ces diagrammes, on peut anticiper les imperfections potentielles et mieux comprendre le comportement attendu de l'application dans divers scénarios. De plus, lors de partage, une communication efficace entre les membres de l'équipe de développement et les parties prenantes est facilitée grâce à une compréhension commune des diagrammes utilisés. En comprenant et en interprétant correctement ces diagrammes, les membres de l'équipe peuvent travailler de manière collaborative pour concevoir et développer une application robuste et fonctionnelle, répondant aux besoins et aux attentes des utilisateurs finaux.

Conclusion générale

Pour conclure, la construction d'un calendrier potager implique la prise en compte de plusieurs facteurs, parmi lesquels le rythme des phases lunaires est largement utilisé, bien que d'autres rythmes puissent également être explorés. Cependant, pour obtenir un calendrier dynamique et précis, le rythme des phases lunaires s'avère être le plus efficace. En utilisant ce rythme, nous avons pu établir les séquences nécessaires à la création d'un calendrier numérique, offrant ainsi une approche complète et précise pour la planification des cultures.

La technologie Android offre une plate-forme accessible à une large gamme d'utilisateurs, qu'ils soient amateurs, professionnels, étudiants ou enseignants. Sa facilité d'utilisation et la conception intuitive de son interface graphique en font un outil idéal pour la création de diverses applications. Le choix d'Android comme plateforme pour notre application de calendrier potager s'est donc révélé évident. Grâce à cette plateforme, nous avons pu développer une application capable d'estimer les différentes plantes pouvant être cultivées au cours d'un mois donné, facilitant ainsi la planification des cultures pour les utilisateurs.

Cependant, ce projet reste en constante évolution et est susceptible d'évoluer rapidement au fil du temps. En tant que perspectives d'amélioration, l'application pourrait être étendue pour inclure l'utilisation d'une base de données externe, permettant à chaque cultivateur de partager les détails de leur jardin et de leurs expériences de culture. Cela transformerait l'application en une sorte de service web, où chacun pourrait commenter, réagir et partager ses propres connaissances et conseils en matière de jardinage. Cette évolution potentielle permettrait à l'application de devenir une ressource collaborative et communautaire pour les passionnés de jardinage, offrant ainsi une expérience enrichie et plus interactive pour tous les utilisateurs.

Liste des acronymes

NL	Nouvelle Lune
PQ	Premier Quartier
PL	Pleine Lune
DQ	Dernier Quartier
CRUD	CReate Update Delete
API	Application Programming Interface.
DVM	Dalvik Virtual Machine.
GUI	Graphic unit interface.
IDE	Integrated Development Environment.
IHM	Interface Homme Machine.
OHA	Open Handset Alliance.
SDK	Software Development Toolkit.
SGBD	Système de Gestion de base de données.
UI	User Interface.
UNA	Université Nord d'Antsiranana

Bibliographie

- [D'I08] Yann D'Isanto. Introduction au sdk android. 2008.
- [RVR⁺00] Alejandro Ramirez, Philippe Vanpeperstraete, Andreas Rueckert, Kunle Odutola, Jeremy Bennett, Linus Tolke, and Michiel van der Wulp. Argouml user manual. 2000.
- [San14] Kartik Sankaran. Android concepts and programming. 2014.
- [wB91] williamann Bell. Astronomical algorithms. 1991.
- [Yad11] Nikhil Yadav. Android development tutorial. 2011.

WEBOGRAPHIE

- 1 <http://www.arcminute-moon-position-algorithm.htm>
- 2 <http://astronomie-smartsmur.over-blog.com/article-4-1-les-periodes-synodiques-88312148.html>
- 3 <https://sites.google.com/site/astronomievueceleste/home>
- 4 <http://www.Agriculturee-biodynamiquee-Definitione-de-Agriculturee-biodynamique-ete-synonymese-de-Agriculturee-biodynamique.htm>
- 5 <http://www.Agro-oi-Union-Matanjaka.htm>
- 6 [http://Astrologie0 quel est votre signe lunaire - Femme Actuelle.html](http://Astrologie0%20quel%20est%20votre%20signe%20lunaire%20-%20Femme%20Actuelle.html)

Annexe A

Annexes

A.1 CODE SOURCE

A.1.1 extrait de modèle de base de donnée

```
1  %code source
2  * Created by ibito-PC on 28/05/2017.
3  */
4  public class Model extends SQLiteOpenHelper {
5      public static final String POTAGER_KEY = "id";
6      public static final String POTAGER_NOM = "nom";
7      public static final String POTAGER_PLANTER = "plante";
8      public static final String POTAGER_TABLE_NAME = "potager";
9
10     public static final String POTAGER_TABLE_CREATE =
11         "CREATE_TABLE_" + POTAGER_TABLE_NAME + "(" + POTAGER_KEY +
12         "_INTEGER_PRIMARY_KEY_AUTOINCREMENT, " +
13         POTAGER_NOM + "_TEXT,_" + POTAGER_PLANTER + "_TEXT)";
14     public static final String POTAGER_TABLE_DROP =
15         "DROP_TABLE_IF_EXISTS_" + POTAGER_TABLE_NAME + ";";
16     public static final String POTAGER_SELECT = "SELECT" +
17         "*" + "_FROM_" + POTAGER_TABLE_NAME + ";";
18     String selectQuery = "Select_name_from_sqlite_master_where_type='table'";
19
20     public Model(Context context, String name,
21         SQLiteDatabase.CursorFactory factory, int version) {
22         super(context, name, factory, version);
23     }
24
25     @Override
26     public void onCreate(SQLiteDatabase db) {
27         db.execSQL(POTAGER_TABLE_CREATE);
28     }
29
30     @Override
31     public void onUpgrade(SQLiteDatabase db,
32         int oldVersion, int newVersion) {
33         db.execSQL(POTAGER_TABLE_DROP);
34         onCreate(db);
35     }
36
37
38
39     public void Sauvegarde(Monpotager m) {
```

```
40     ContentValues value = new ContentValues();
41     value.put(POTAGER_NOM, m.getNom());
42     value.put(POTAGER_PLANTER, m.getPlante());
43     Long id = m.getId();
44     if (id == 0) {
45
46         id = this.getWritableDatabase().
47             insert(POTAGER_TABLE_NAME, null, value);
48         m.setId(id);
49
50     } else {
51         this.getWritableDatabase().update(POTAGER_TABLE_NAME,
52             value, POTAGER_KEY + "=?", new String[]{String.valueOf(id)});
53     }
54
55 }
56
57 public void effacertout() {
58     this.getWritableDatabase().delete(POTAGER_TABLE_NAME,
59         null, null);
60 }
61
62 public void effacerPotager(Monpotager p) {
63     this.getWritableDatabase().delete(POTAGER_TABLE_NAME,
64         POTAGER_KEY + "=?", new String[]{String.valueOf(p.getId())});
65 }
66
67 public Cursor getAllpotagerAscursor() {
68     return this.getReadableDatabase().rawQuery(POTAGER_SELECT, null);
69 }
70 public List<String> allTableName() {
71     List<String> result = new ArrayList<String>();
72     Cursor c = this.getReadableDatabase().rawQuery(selectQuery, null);
73     if (c.moveToFirst()) {
74         do {
75             String n = c.getString(c.getColumnIndex("name"));
76             result.add(n);
77         } while (c.moveToNext());
78     }
79     return result;
80 }
81 public List<Monpotager> getAllpotager() {
82     List<Monpotager> resultat = new ArrayList<Monpotager>();
83     Cursor c = this.getAllpotagerAscursor();
84     /*
85     if (c.moveToFirst()) {
86         do {
87             Long id = c.getLong(c.getColumnIndex(POTAGER_KEY));
88             p.setId(id);
89             p.setNom(c.getString(c.getColumnIndex(POTAGER_NOM)));
90             p.setPlante(c.getString(c.getColumnIndex(POTAGER_PLANTER)));
91             resultat.add(p);
92         } while (c.moveToNext());
93     }
94
95     if (c.getCount() == 0) {
96         return null;
97     } else {
98         c.moveToFirst();
```

```
99         while (c.isAfterLast() == false) {
100             Monpotager p = new Monpotager();
101             Long id = c.getLong(c.getColumnIndex(POTAGER_KEY));
102             p.setId(id);
103             p.setNom(c.getString(c.getColumnIndex(POTAGER_NOM)));
104             p.setPlante(c.getString(c.getColumnIndex(POTAGER_PLANTER)));
105             Log.i("test_BD", "liste "+c.getString(c.getColumnIndex(POTAGER_NOM)));
106             resulat.add(p);
107             c.moveToNext();
108         }
109         c.close();
110         return resulat;
111     }
112 }
113 }
```

A.1.2 extrait de code pour lister les potager

```
1
2 /**
3  * Created by ibito-PC on 25/05/2017.
4  */
5 public class Liste_potagerActivity extends AppCompatActivity {
6     //liste
7     ArrayList liste_potager;
8     private ListView liste = null;
9     ArrayAdapter<Potager> adapter;
10    Resources resources_potager;
11    final String EXTRA_POTAGE="potage";
12    public Liste_potagerActivity(){
13    }
14
15    public void onCreate(Bundle savedInstanceState){
16        super.onCreate(savedInstanceState);
17        //setContent(R.layout.activity_liste);
18        //creation pop up
19        AlertDialog.Builder alertDialog =
20        new AlertDialog.Builder(Liste_potagerActivity.this);
21        LayoutInflater inflater = getLayoutInflater();
22        View convertView = (View) inflater.inflate(R.layout.custom, null);
23        alertDialog.setView(convertView);
24        alertDialog.setTitle("List_de_potage");
25
26        //lecture de resource
27        resources_potager =getResources();
28        final String[] nom_potager = resources_potager.getStringArray(R.array.potage);
29        //recopie dans le ArrayList
30        liste_potager = new ArrayList<Potager>();
31        for (int i=0; i<nom_potager.length; ++i) {
32            liste_potager.add(new Potager(nom_potager[i]));
33        }
34        //adapteur
35        adapter = new ArrayAdapter<Potager>(this ,
36        android.R.layout.simple_list_item_1 , liste_potager);
37        //liste= (ListView) findViewById(R.id.listView2);
38
39
40        ListView lv = (ListView) convertView.findViewById(R.id.listView1);
```

```
41         lv.setAdapter(adapter);
42         lv.setOnItemClickListener(new AdapterView.OnItemClickListener()
43             {
44
45             @Override
46             public void onItemClick(AdapterView<?> parent, View view, int position,
47                                     long id) {
48                 // String content = (String)parent.
49                 getItemAtPosition(position);
50                 // Log.i("potage","-----");
51                 Intent intent=new Intent(Liste_potagerActivity.this,
52                     Choix_potageActivity.class);
53                 intent.putExtra(EXTRA_POTAGE, liste_potager.get(position).toString());
54                 startActivity(intent);
55                 finish();
56
57             }
58
59         });
60         alertDialog.show();
61     }
62     public void onItemClick(ListView l, View v, int position, long id) {
63     }
64 }
65 public void onBackPressed() {
66     super.onBackPressed();
67     finish();
68 }
69
70 }
```

A.1.3 calcul de Phase Lunaire

```
1
2 /**
3  * Created by ibito-PC on 07/05/2017.
4  */
5 public class Calcul{
6     private double NL,PL,PQ,DQ;
7     private int nomMois,nomAnnee;
8     private boolean resultat;
9     private ArrayList<Mois> moisListe;
10    int m;
11
12    public Calcul(int mois,int annee,Context context) {
13        int [] nJour= context.getResources().getIntArray(R.array.nJour);
14        String [] noms = context.getResources().getStringArray(R.array.noms);
15        nomMois=mois;
16        nomAnnee=annee;
17        resultat=bissextille(nomAnnee);
18        //creation du liste
19        moisListe = new ArrayList<Mois>();
20
21        for (int i=0; i<noms.length; ++i) {
22            moisListe.add(new Mois(noms[i],nJour[i],i));
23        }
24        //changement de fevrier
25        if(nomMois==2 && resultat==true){
```

```
26         moisListe.remove(1);
27         moisListe.add(1,new Mois("Fevrier",29,1));
28     }
29     else{
30         moisListe.remove(1);
31         moisListe.add(1,new Mois("Fevrier",28,1));
32     }
33     NL=calcul(mois,resultat,nomAnnee,"NL");
34     PL=calcul(mois,resultat,nomAnnee,"PL");
35     PQ=calcul(mois,resultat,nomAnnee,"PQ");
36     DQ=calcul(mois,resultat,nomAnnee,"DQ");
37 }
38 private double calcul(int m,boolean bis,int annee,String phase_lune){
39     float k,nbrjour_annee,somme=0,T,E,M,M_second,F,n,somme_cor,somme_cor2,Z;
40     float[] tab_cor=new float[26];
41     double JD,JDE,reste;
42     int alpha,A,B,C,D,E_final,reponse;
43
44     for(int i=m;i>=0;--i)
45         somme=somme+moisListe.get(i).getNbrJour();
46     if(bis==true)
47         nbrjour_annee=366;
48     else
49         nbrjour_annee=365;
50     somme=somme/nbrjour_annee;
51     somme=(float)arrondi(somme,2);
52     somme=somme+annee;
53     k=calcul_k(somme,phase_lune);
54     T=(float)arrondi(k/1236.85,5);
55     JDE=(double)(2451550.09765+(29.530588853*k)+(0.0001337*T*T)-(0.00000015*T*T*T)+(0.00000000
56     E=(float)(1-(0.002516*T)-(0.0000074*T*T));
57     M=(float)(2.5534 + (29.10535669*k)-(0.0000218*T*T)-(0.00000011*T*T*T));
58     M_second=(float)(201.5643 + (385.81693528 *(k)) + (0.0107438*T*T) + (0.00001239*T*T*T) -
59     F=(float)(160.7108 + 390.67050274*(k) - 0.0016341*T*T - 0.00000227*T*T*T + 0.000000011*T*
60     n=(float)(124.7746 - 1.5637558*(k) + 0.0020691*T*T + 0.00000215 *T*T*T);
61
62     tab_cor[0]=(float)(299.77 + 0.107408*k - 0.009173*T*T);
63     tab_cor[1]=(float)(251.88 + 0.016321*k);
64     tab_cor[2]=(float)(251.83 + 26.651886*(float)k);
65     tab_cor[3]=(float)(349.42 + 36.412478*k);
66     tab_cor[4]=(float)( 84.66 + 18.206239*k);
67     tab_cor[5]=(float)(141.74 + 53.303771*k);
68     tab_cor[6]=(float)(207.14 + 2.453732*k);
69     tab_cor[7]=(float)(154.84 + 7.30686*k);
70     tab_cor[8]=(float)(34.52 + 27.261239*k);
71     tab_cor[9]=(float)(207.19 + 0.121824 *k);
72     tab_cor[10]=(float)(291.34 + 1.844379 *k);
73     tab_cor[11]=(float)(161.72 + 24.198154*k);
74     tab_cor[12]=(float)(239.56 + 25.513099 *k);
75     tab_cor[13]=(float)(331.55 + 3.592518*k);
76
77     tab_cor[0]=(float)(0.000325*Math.sin(Math.toRadians(tab_cor[0])));
78     tab_cor[1]=(float)(0.000165*Math.sin(Math.toRadians(tab_cor[1])));
79     tab_cor[2]=(float)(0.000164*Math.sin(Math.toRadians(tab_cor[2])));
80     tab_cor[3]=(float)(0.000126*Math.sin(Math.toRadians(tab_cor[3])));
81     tab_cor[4]=(float)(0.00011*Math.sin(Math.toRadians(tab_cor[4])));
82     tab_cor[5]=(float)(0.000062*Math.sin(Math.toRadians(tab_cor[5])));
83     tab_cor[6]=(float)(0.00006*Math.sin(Math.toRadians(tab_cor[6])));
84     tab_cor[7]=(float)(0.000056*Math.sin(Math.toRadians(tab_cor[7])));
```

```
85     tab_cor[8]=(float)(0.000047*Math.sin(Math.toRadians(tab_cor[8])));
86     tab_cor[9]=(float)(0.000042*Math.sin(Math.toRadians(tab_cor[9])));
87     tab_cor[10]=(float)(0.00004*Math.sin(Math.toRadians(tab_cor[10])));
88     tab_cor[11]=(float)(0.000037*Math.sin(Math.toRadians(tab_cor[11])));
89     tab_cor[12]=(float)(0.000035*Math.sin(Math.toRadians(tab_cor[12])));
90     tab_cor[13]=(float)(0.000023*Math.sin(Math.toRadians(tab_cor[13])));
91
92     somme_cor=0;
93     for(int i=0;i<=13;++i)
94         somme_cor=somme_cor+tab_cor[i];
95
96     tab_cor[0]=(float)(-0.4072*Math.sin(Math.toRadians(M_second)));
97     tab_cor[1]=(float)(0.17241*E*Math.sin(Math.toRadians(M)));
98     tab_cor[2]=(float)(0.01608*Math.sin(Math.toRadians(2*M_second)));
99     tab_cor[3]=(float)(0.01039*Math.sin(Math.toRadians(2*F)));
100    tab_cor[4]=(float)(0.00739*E*Math.sin(Math.toRadians(M_second-M)));
101    tab_cor[5]=(float)(-0.00514*E*Math.sin(Math.toRadians(M_second+M)));
102    tab_cor[6]=(float)(0.00208*E*E*Math.sin(Math.toRadians(2*M)));
103    tab_cor[7]=(float)(-0.00111*Math.sin(Math.toRadians(M_second-2*F)));
104    tab_cor[8]=(float)(-0.00057*Math.sin(Math.toRadians(M_second+2*F)));
105    tab_cor[9]=(float)(0.00056 *Math.sin(Math.toRadians(2*M_second+M)));
106    tab_cor[10]=(float)(-0.00042*Math.sin(Math.toRadians(3*M_second)));
107    tab_cor[11]=(float)(0.00042 *Math.sin(Math.toRadians(M+2*F)));
108    tab_cor[12]=(float)(0.00038*E*Math.sin(Math.toRadians(M-2*F)));
109    tab_cor[13]=(float)(-0.00024*E*Math.sin(Math.toRadians(2*M_second-M)));
110    tab_cor[14]=(float)(-0.00017*Math.sin(Math.toRadians(n)));
111    tab_cor[15]=(float)(-0.00007*Math.sin(Math.toRadians(M_second-2*M)));
112    tab_cor[16]=(float)(0.00004*Math.sin(Math.toRadians(2*M_second-2*F)));
113    tab_cor[17]=(float)(0.00004*Math.sin(Math.toRadians(3*M)));
114    tab_cor[18]=(float)(0.00003*Math.sin(Math.toRadians(M_second+M-2*F)));
115    tab_cor[19]=(float)(0.00003*Math.sin(Math.toRadians(2*M_second+2*F)));
116    tab_cor[20]=(float)(-0.00003*Math.sin(Math.toRadians(M_second+M+2*F)));
117    tab_cor[21]=(float)(0.00003*Math.sin(Math.toRadians(M_second-M+2*F)));
118    tab_cor[22]=(float)(-0.00002*Math.sin(Math.toRadians(M_second-M-2*F)));
119    tab_cor[23]=(float)(-0.00002*Math.sin(Math.toRadians(3*M_second-M)));
120    tab_cor[24]=(float)(0.00002*Math.sin(Math.toRadians(4*M_second)));
121
122    somme_cor2=0;
123    for(int i=0;i<=24;++i)
124        somme_cor2=somme_cor2+tab_cor[i];
125
126    JD=JDE+somme_cor2+somme_cor;
127    JD=JD+0.5;
128    Z=(int)JD;
129    reste=JD-Z;
130    reste=arrondi(reste,5);
131    alpha=(int)((Z-1867216.25)/36524.25);
132    A=(int)(Z+1+alpha-(alpha/4));
133    B=A+1524;
134    C=(int)((B-122.1)/365.25);
135    D=(int)(365.25*C);
136    E_final=(int)((B-D)/30.6001);
137    reponse=(B-D-(int)(30.6001*E_final)+(int)reste);
138    return reponse;
139 }
140
141 public static double arrondi(double A, int B) {
142     return (double) ( (int) (A * Math.pow(10, B) + .5)) / Math.pow(10, B);
143 }
```



```
144     public static float calcul_k(float somme, String phase_lune){
145         float k=0;
146         k=(int)((somme-2000)*12.3685);
147         if(phase_lune=="NL")
148             k=(float)(k+0);
149         else if(phase_lune=="PQ"){
150             k=(float)(k+0.25);
151         }
152         else if(phase_lune=="DQ")
153             k=(float)(k+0.75);
154         else
155             k=(float)(k+0.5);
156
157         return k;
158     }
159     //bisextille
160     private boolean bissextille(int annee){
161         boolean resultat;
162         if((annee%400)==0||((annee%100!=0)&&(annee%4==0)))
163             resultat=true;
164         else
165             resultat=false;
166         return resultat;
167     }
168
169
170 }
```

A.1.4 choix des potager

```
1         package calendrierpotager.ibito.com.calendrierpotager;
2
3     import android.content.Context;
4     import android.content.Intent;
5     import android.content.SharedPreferences;
6     import android.os.Bundle;
7     import android.preference.PreferenceManager;
8     import android.support.v7.app.AppCompatActivity;
9     import android.util.Log;
10    import android.widget.CalendarView;
11    import android.widget.TextView;
12    import android.widget.Toast;
13
14    /**
15     * Created by ibito-PC on 19/04/2017.
16     */
17
18    public class CalendrierActivity extends AppCompatActivity implements CalendarView.OnDateChangeListener {
19        final String EXTRA_JOUR="jour";
20        final String EXTRA_MOIS="mois";
21        final String EXTRA_ANNEE="annee";
22        final String TYPE_PLANTE="type_plante";
23        //sesion
24        String myString,typepotage;
25        Calcul cal_calend;
26        int jour, mois, annee, NL, PL, DQ, PQ;
27        public static final String MyPREFERENCES = "Mon_reference" ;
28        public static final String Name = "nameKey";
```

```
29
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_calendrier);
33         // session
34         SharedPreferences preferences = getSharedPreferences(MyPREFERENCES, 0);
35         myString = preferences.getString(Name, null);
36         //Log.i("information","-----");
37         //affichage textView5
38         TextView t1=(TextView) findViewById(R.id.textView7);
39         TextView t2 = (TextView) findViewById(R.id.textView5);
40         if (myString==null){
41             t1.setText("");
42             t2.setText("");
43         }
44         else{
45             t2.setText(myString);
46         }
47         CalendarView calendrier =(CalendarView) findViewById(R.id.calendarView);
48         calendrier.setOnDateChangeListener(this);
49     }
50
51     @Override
52     public void onSelectedDayChange(CalendarView view, int year, int month, int dayOfMonth) {
53         if (myString==null){
54
55             cal_calend = new Calcul(month, year, getApplicationContext());
56             NL = (int) cal_calend.getNL();
57             PL = (int) cal_calend.getPL();
58             PQ = (int) cal_calend.getDQ();
59             DQ = (int) cal_calend.getPQ();
60             if (dayOfMonth<PQ)
61                 typepotage="feuille";
62             else if (dayOfMonth>PQ && dayOfMonth<NL)
63                 typepotage="fleur";
64             else if (dayOfMonth>PQ && dayOfMonth<NL)
65                 typepotage="fruit";
66             else
67                 typepotage="racine";
68
69             // Liste_potagerActivity l=new Liste_potagerActivity();
70             Intent intent=new Intent(CalendarActivity.this, Liste_potagerActivity.class);
71             intent.putExtra(TYPE_PLANTE, typepotage);
72             this.startActivity(intent);
73         }
74         else{
75             Intent intent=new Intent(CalendarActivity.this, Choix_potageActivity.class);
76             intent.putExtra(EXTRA_JOUR, ""+dayOfMonth);
77             intent.putExtra(EXTRA_MOIS, ""+month);
78             intent.putExtra(EXTRA_ANNEE, ""+year);
79             Toast.makeText(CalendarActivity.this, "jour"+dayOfMonth, Toast.LENGTH_SHORT).show();
80             this.startActivity(intent);
81         }
82     }
83
84     public void onBackPressed() {
85         SharedPreferences sharedPreferences = getSharedPreferences(CalendarActivity.MyPREFERENCES, 0);
86         SharedPreferences.Editor editor = sharedPreferences.edit();
87         editor.clear();
```

```
88         editor.commit();
89         super.onBackPressed();
90         finish();
91     }
92 }
```

