PROGRAMME DETAILLÉ DU JEUDI 13 MAI 2019

	Titane 2	Chrome 1	Chrome 2 + 3	Chrome 4
8h00	ACCUEIL			
9h00	**Equiper sa voie** Thomas Pierrain KEYNOTE EN SALLE TITANE 2: Au-delà de la passion, faire du logiciel pour moi c'est aider les autres à résoudre leurs problèmes. Comment aider les autres si on n'est pas déjà clair soi-même avec sa posture, son métier, ses envies, ses ambitions ou bien ses craintes ? C'est la question que je me suis posé à plusieurs reprises durant ma carrière. Et c'est de cela - et des quelques aides trouvées en route - dont je voudrais vous parler.			
10h00	PAUSE EN SALLE TITANE 1			
10h10	**Petite introduction au TDD par l'exemple** Xavier Nopre Pour faire du développement itératif et incrémental, nous devons faire évoluer leur code en permanence. Pour cela, il est indispensable d'avoir des tests unitaires, et la meilleure façon de les écrire, c'est le TDD, qui présente bien d'autres avantages ! Après un peu de théorie, je ferai une démo en live coding pour illustrer cette pratique toujours trop peu utilisée.	**Designé pour être jeté!** Bastien David Au début de sa carrière, on tire une certaine fierté de la quantité de code écrit. Au fil du temps, on prend de plus en plus de fierté à jeter du code, si bien que certains développeurs se vantent parfois d'avoir supprimé plus de code qu'ils en ont écrit sur une journée.	**Découvrez l'Example mapping par la pratique** Thomas Pierrain ATELIER: Le BDD est trop souvent réduit à de l'outillage, alors que c'est surtout un super moyen de rapprocher le métier et les professionnels du software pour livrer des applications qui font ce dont on a réellement besoin. Malgré sa redoutable efficacité, L'Example Mapping n'est pas encore beaucoup utilisé et demeure plutôt méconnu. Après une rapide introduction à l' Example Mapping nous le pratiquerons ensemble autours d'un cas concret.	**Pimper sa chaîne de build** Laurent Tardif ATELIER: Installons une chaîne de build complète sur son portable et voir comment interagissent chaque composant de cette forge. Nous discuterons aussi de quels sont les pièges à éviter.
11h00				
11h10	**Du legacy au TDD** Johan Martinsson C'est quoi l'obstacle principal à travailler avec les tests ou en TDD. C'est que le code existant n'a pas été concu pour! Voyons à travers un exemple comment on reprend le code, le prépare au travail en TDD à l'aide des tests :) et du refactoring préparatoire afin que cela devienne un jeu d'enfant d'ajouter la nouvelle fonctionnalité en TDD (ou presque :D)	**La revue de code, sur le terrain** Manuel Vacelet La revue de code est un art difficile, et contrairement à ce que l'on pourrait croire, ça ne va pas en s'arrangeant avec le temps comme par magie. Cette présentation est un état des leçons apprises sur les 8 années de revue de code pratiquées sur le projet Tuleap.		**Démarrer vite, tester, apprendre, partager et recommencer !** David Crosson Nous allons vivre une approche itérative et apprenante du développement basé sur ammonite et scala. Cette approche est basée sur la création d'exemples très facilement exécutable et partageable via les gists de github ou les snippets de gitlab. La session se fera essentiellement en live coding
12h00				
12h00	DEJEUNER EN SALLE TITANE 1			
13h30	**Des principes de design à la rescousse des microservices** Jérôme Avoustin Les architectures basées sur les microservices ont le vent en poupe. Tout le monde veut en faire. Les promesses en terme de bénéfice sont très séduisantes. Mais implémenter ce type d'architecture n'est pas sans difficulté, c'est peu de le dire. On risque en effet à un monolithe distribué ...	**Apprendre et Transmettre le savoir craft** Houssam Fakih Ce talk est un retour d'expérience sur la mise en place d'un training craft de 4 semaines. Son objectif principal est de transmettre des méthodes, des techniques, des connaissances et un savoir-faire craft à d'autres développeurs. Nous traiterons de tous les aspects liés au training : la préparation, le format, le contenu, l'animation, le suivi, l'amélioration.	**Ping pong pair programming** Nastasia Saby ATELIER : Il s'agit de l'association de 2 pratiques : le pair programming (programmation en binôme) et le TDD (Test Driven Development). Cette pratique qui reprend le rythme du jeu du ping pong est l'occasion de découvrir, tester, expérimenter, progresser, rendre plus pertinents le pair programming, le TDD et ainsi notre pratique quotdienne du développement.	**Maîtriser son IDE** Rémy Sanlaville ATELIER: Tout bon artisan se démarque dans la maîtrise de ses outils. En tant que développeur, l'IDE est votre outil quotidien principal. Bien le maîtriser est un atout indéniable pour à aller plus vite et vous concentrer sur l'essentiel. Cela vous apportera rapidité, fluidité et plaisir de coder. Venez vous entraîner sur votre outil préféré (IntelliJ ou Eclipse) afin d'apprendre des fonctionnalités et des astuces insoupçonnées.
14h20				
14h30	**D'impératif au fonctionnel** Patrick Giry Comment faire de la programmation fonctionnelle quand on a les pieds liés au code impératif et mutable	**Méthodes UX faciles, ou comment l'artisan s'assure qu'il va contenter ses clients** Nathalie Cotte et Romain Bioteau La UX résout un constat évident : le point de vue du client sur un produit n'est pas le même que celui de l'artisan qui l'a conçu. Pourtant il faut concilier ces points de vue pour transformer l'artiste en artisan. Il produit ainsi un très beau produit, qui ravit les clients et lui assure sa pérennité. Nous parlerons donc de la valeur perçue et nous évoquerons en quoi la collaboration Dev / UX ne peut qu'être fructueuse... à quelques conditions près.		
15h20				
15h20	PAUSE EN SALLE TITANE 1			
15h40	**Exploration de différents styles d'architecture** Clément Bouillier On entend souvent parler d'architecture en couche/n-tiers, d'architecture hexagonale/en onion, de patterns divers et variés (Repository, DAO, Active Record, Domain Model anémique ou pas, ORM...), de DDD, de CQRS et d'EventSourcing...ce talk en fait un tour d'horizon sous la forme d'une revue de code de différentes architectures d'un même exemple.	**Metrics Driven Development** Pierre Chiron Vous vous êtes un jour douté de la pertinence des features proposés? Et si on collectait un peu de données pour s'assurer qu'on développait des fonctionnalités que les utilisateurs utilisent réellement?	**Carpaccio d'éléphant** Christophe Maldivi et Emeric Fontaine ATELIER: Apprenez à découper les users stories de votre produit en tranches très fines, faites un carpaccio d'éléphant !	**Open Closed Principle - Le challenge** Mathieu Cans ATELIER: Dans cette session, nous apprendrons ensemble à pratiquer la pierre angulaire des principes S.O.L.I.D. : le Open / Close principle. Nous travaillerons en mob programming ce qui nous contraindra à verbaliser les solution et permettra à chacun d' être actif. Nous modifierons légèrement le flux du TDD pour nous contraindre à respecter le principe Open/Close. Nous essaierons aussi de supprimer tous les if de notre code.
16h30				
16h40	**En finir avec la «Dette Technique»** Christophe Thibaut Au delà de subir la dette technique comment pouvons nous trouver le dialogue qui permettra de commencer à en sortir? Ward Cunningham a nommé "Technical Debt" le procédé qui consiste, pour une équipe, à déroger temporairement à ses standards de qualité en vue de gagner du temps dans la réalisation d'un objectif intermédiaire. Au cours de la dernière décennie, l'expression est devenue très populaire, tout en perdant son sens initial. Elle est désormais utilisée pour désigner l'état d'une solution jugée non conforme à l'état de l'art généralement admis dans notre industrie, et pour souligner le fait que c'est cette non-qualité qui ralentit la progression de l'équipe aux prise avec cette solution. La plupart des équipes de développement ne contractent pas délibérément de dette technique. La plupart des équipes subissent de la dette technique. Si pour résoudre un problème il faut d'abord le définir aussi clairement que possible alors nous devons reconnaître que le terme "Dette Technique" ne contribue en rien à clarifier l'état de ce qu'il est censé décrire. Nous devons donc trouver un modèle alternatif qui nous donne une meilleure façon de parler de cet état des choses que nous appelons une solution "endettée" si nous voulons contribuer à l'améliorer. C'est ce que je me propose de faire durant cette présentation.	**Des callbacks à async/await : une histoire asynchrone** Jerome Avoustin Pendant longtemps, JavaScript s'en est tenu au principe de callbacks pour la gestion des appels asynchrones. Dans son histoire récente, JS s'est armé pour simplifier l'écriture d'un code qui s'éxecute de manière asynchrone, en lui donnant un aspect synchrone, grâce à async/await. Mais comment en est-on arrivé là ? Quelles avancées ont permis d'obtenir cela ? Nous verrons cela lors d'une session de live coding pour revenir ou présenter quelques concepts clés de JS, comme les promesses, ou la plus belle évolution de JS depuis sa création : les générateurs.		
17h30				
17h40	CLOTURE ET APERO			
18h00				