PERANGKAT LUNAK TANGKAS

Teknologi yang selalu berubah​

Komunikasi yang baik antar tim dan klien​

Pengelolaan waktu​

Kompleksitas proyek​

Keamanan ​

Pengujian yang bebas dari bug​

Pemeliharaan secara berkala​

**Tantangan Developer dalam Pengembangan Perangkat Lunak Tradisional**

Sebelum era Agile, pengembangan perangkat lunak sering kali dihadapkan dengan berbagai tantangan, seperti:

.**Kurangnya komunikasi dan kolaborasi:** Tim pengembangan sering kali terkotak-kotak, dengan sedikit komunikasi dan kolaborasi antar departemen. Hal ini dapat menyebabkan kesalahpahaman, keterlambatan, dan produk akhir yang tidak sesuai dengan kebutuhan pengguna.

.**Proses yang kaku dan tidak fleksibel:** Proses pengembangan perangkat lunak tradisional umumnya kaku dan tidak fleksibel, dengan sedikit ruang untuk perubahan. Hal ini dapat membuat sulit untuk beradaptasi dengan perubahan kebutuhan pengguna atau pasar.

.**Fokus pada dokumentasi daripada fungsionalitas:** Perhatian sering kali lebih difokuskan pada pembuatan dokumentasi yang ekstensif daripada pada pengembangan perangkat lunak yang fungsional dan bermanfaat bagi pengguna.

.**Pengujian yang terlambat:** Pengujian sering kali dilakukan di akhir proses pengembangan, yang dapat menyebabkan cacat yang mahal dan memakan waktu untuk diperbaiki.

.**Kurangnya umpan balik pengguna:** Pengguna sering kali tidak terlibat dalam proses pengembangan, yang dapat menyebabkan produk akhir yang tidak sesuai dengan kebutuhan mereka

-**Mengapa Agile?**

Metodologi Agile dikembangkan untuk mengatasi tantangan-tantangan ini dengan menyediakan pendekatan yang lebih adaptif, kolaboratif, dan berpusat pada pengguna untuk pengembangan perangkat lunak. Berikut beberapa alasan mengapa Agile menjadi pilihan populer:

.**Meningkatkan komunikasi dan kolaborasi:** Agile mendorong komunikasi dan kolaborasi yang konstan antara tim pengembangan, pemangku kepentingan, dan pengguna. Hal ini dapat membantu mengurangi kesalahpahaman, meningkatkan kualitas produk, dan memastikan bahwa produk akhir memenuhi kebutuhan semua pihak.

.**Proses yang fleksibel dan adaptif:** Agile memungkinkan perubahan dilakukan dengan mudah dan cepat selama proses pengembangan. Hal ini memungkinkan tim untuk beradaptasi dengan perubahan kebutuhan pengguna atau pasar tanpa mengganggu alur kerja.

.**Fokus pada fungsionalitas dan nilai pengguna:** Agile menekankan pengembangan perangkat lunak yang fungsional dan bermanfaat bagi pengguna. Hal ini dicapai dengan melibatkan pengguna dalam proses pengembangan dan sering kali meminta umpan balik mereka.

.**Pengujian yang berkelanjutan:** Pengujian diintegrasikan ke dalam proses pengembangan Agile, yang berarti cacat dapat diidentifikasi dan diperbaiki lebih awal. Hal ini dapat menghemat waktu dan uang dalam jangka panjang.

.**Peningkatan kepuasan pengguna:** Agile membantu memastikan bahwa produk akhir memenuhi kebutuhan pengguna, yang mengarah pada peningkatan kepuasan pengguna.

-**Sejarah Agile**

**.**Metodologi Agile muncul pada awal tahun 2000-an sebagai reaksi terhadap metode pengembangan perangkat lunak tradisional yang kaku dan tidak fleksibel. Pada tahun 2001, sekelompok pengembang perangkat lunak berkumpul untuk membuat Manifesto Agile Software Development, yang menguraikan nilai-nilai dan prinsip-prinsip dasar Agile. Sejak saat itu, berbagai metodologi Agile telah dikembangkan, seperti Scrum, Kanban, dan Extreme Programming (XP)

**Agile 101: Pengenalan Singkat**

**Agile** adalah pendekatan pengembangan perangkat lunak yang fleksibel dan adaptif. Ini diciptakan untuk mengatasi masalah metode lama yang kaku dan tidak bisa menyesuaikan diri dengan perubahan. Agile lebih mementingkan kolaborasi, komunikasi, dan kepuasan pengguna.

**Komponen Penting Agile:**

**Agile Manifesto:**

.Dokumen yang mendeklarasikan nilai-nilai inti Agile, seperti individu dan interaksi lebih penting daripada proses dan alat.

.Menekankan pentingnya kolaborasi, perangkat lunak yang berfungsi, dan kemampuan beradaptasi dengan perubahan.

## Contoh Prinsip Agile Manifesto:

**Individu dan interaksi** lebih penting daripada proses dan alat.

**Perangkat lunak yang bekerja** lebih penting daripada dokumentasi yang komprehensif.

**Kolaborasi dengan pelanggan** lebih penting daripada negosiasi kontrak.

**Menanggapi perubahan** lebih penting daripada mengikuti sebuah rencana.

**Agile Manifesto dan Prinsip-Prinsipnya**

**Agile Manifesto** adalah dokumen yang ditulis oleh 17 pengembang perangkat lunak independen pada tahun 2001. Manifesto ini berisi empat nilai inti dan 12 prinsip yang mendasari pengembangan perangkat lunak Agile.

**Nilai Inti Agile Manifesto:**

**.Individu dan interaksi lebih dari proses dan alat.**

**.Software yang bekerja lebih dari dokumentasi yang komprehensif.**

**.Kolaborasi pelanggan lebih dari negosiasi kontrak.**

**.Merespons perubahan lebih dari mengikuti rencana.**

**Prinsip-Prinsip Agile Manifesto:**

1. **Prioritaskan kepuasan pelanggan dengan memberikan pengiriman perangkat lunak berkelanjutan secara berkala dan awal.**
2. **Sambut perubahan persyaratan, bahkan pada tahap akhir pengembangan. Agile memungkinkan penyesuaian dengan perubahan secara efektif.**
3. **Kirimkan perangkat lunak yang berfungsi secara berkala, dengan preferensi dalam siklus waktu singkat dan sering.**
4. **Pengembang dan bisnis harus bekerja sama setiap hari.**
5. **Bangun proyek di sekitar individu yang termotivasi. Berikan mereka ruang, kepercayaan, dan dukungan yang mereka butuhkan.**
6. **Metode komunikasi yang paling efektif adalah percakapan tatap muka.**
7. **Software yang bekerja adalah ukuran utama kemajuan.**
8. **Pertahankan ritme yang berkelanjutan. Tim Agile dapat mempertahankan fokus dan energi yang tinggi dalam jangka waktu yang lama.**
9. **Perhatian terus-menerus terhadap keunggulan teknis dan desain yang baik meningkatkan kelincahan.**
10. **Kesederhanaan - seni memaksimalkan jumlah pekerjaan yang tidak dilakukan - sangat penting.** 11. **Tim terbaik menghasilkan diri mereka sendiri dengan disiplin.**
11. **Secara berkala refleksikan bagaimana tim dapat menjadi lebih efektif dan lakukan penyesuaian yang diperlukan.**

**Contoh Penerapan Agile Manifesto:**

* **Scrum:** Scrum adalah kerangka kerja Agile populer yang membagi proyek menjadi sprint singkat, dengan fokus pada kolaborasi, adaptasi, dan penyampaian nilai secara berulang.
* **Kanban:** Kanban adalah metode visual untuk mengelola alur kerja, dengan fokus pada batasan pekerjaan yang sedang berlangsung (WIP) dan aliran yang lancar.
* **Extreme Programming (XP):** XP adalah metodologi Agile yang menekankan pada pengujian unit yang berkelanjutan, pemrograman berpasangan, dan integrasi berkelanjutan.

**Manfaat Agile Manifesto:**

* **Peningkatan kualitas perangkat lunak:** Agile menekankan pada pengiriman perangkat lunak yang sering dan umpan balik pelanggan, yang membantu untuk mengidentifikasi dan memperbaiki bug dengan cepat.
* **Peningkatan kepuasan pelanggan:** Agile berfokus pada memenuhi kebutuhan pelanggan dan melibatkan mereka dalam proses pengembangan, yang mengarah pada peningkatan kepuasan pelanggan.
* **Peningkatan kelincahan:** Agile memungkinkan tim untuk beradaptasi dengan perubahan dengan mudah dan merespons kebutuhan baru dengan cepat.
* **Peningkatan motivasi tim:** Agile menekankan pada kolaborasi, kepemilikan, dan komunikasi, yang dapat meningkatkan motivasi tim dan meningkatkan kinerja.

**Agile Manifesto** telah memberikan pengaruh yang signifikan pada pengembangan perangkat lunak dan telah diadopsi oleh banyak organisasi di seluruh dunia. Prinsip-prinsipnya yang berfokus pada nilai, kolaborasi, dan adaptasi dapat membantu tim untuk mengembangkan perangkat lunak berkualitas tinggi yang memenuhi kebutuhan pelanggan.

**Kesimpulan:**

Agile Manifesto adalah panduan berharga untuk pengembangan perangkat lunak modern. Nilai-nilai dan prinsip-prinsipnya membantu tim untuk mengembangkan perangkat lunak berkualitas tinggi yang memenuhi kebutuhan pelanggan dengan cara yang lincah dan adaptif.

**12 Prinsip Agile:**

Prinsip-prinsip ini menjelaskan bagaimana menerapkan nilai-nilai Agile dalam praktik.

Contoh prinsip:

Prioritas tertinggi adalah memuaskan pengguna dengan pengiriman perangkat lunak yang bernilai secara terus menerus.

Merangkul perubahan kebutuhan, bahkan terlambat dalam pengembangan.

**Metode-Metode Agile:**

.Ada berbagai metode pengembangan yang mengikuti prinsip Agile, seperti Scrum, Kanban, dan Extreme Programming (XP).

.Setiap metode memiliki ciri khas tersendiri, namun semuanya menekankan fleksibilitas, kolaborasi, dan pengiriman nilai cepat kepada pengguna.

**Intinya:** Agile adalah cara yang lebih responsif dan adaptif untuk mengembangkan perangkat lunak. Ini membantu tim untuk bekerja sama secara lebih efektif, menghasilkan produk yang lebih baik, dan beradaptasi dengan perubahan dengan lebih mudah.

**Alat Pendukung Implementasi Agile:**

**Pengenalan Alat Pendukung Agile:**

Banyak alat tersedia untuk membantu tim menerapkan metodologi Agile.

-Alat ini dapat membantu meningkatkan kolaborasi, melacak kemajuan, dan mengelola proyek Agile.

-Beberapa kategori umum alat Agile termasuk:

**.Alat Manajemen Papan Kanban:** Memvisualisasikan alur kerja dan membantu tim melacak kemajuan.

**.Alat Pelacakan Bug:** Melacak dan mengelola cacat dan masalah perangkat lunak.

**.Alat Manajemen Sprint:** Membantu tim merencanakan, melacak, dan menyelesaikan sprint Agile.

**.Alat Kolaborasi:** Memfasilitasi komunikasi dan kolaborasi antar anggota tim.

**Penerapan Git untuk Kolaborasi:**

* Git adalah sistem kontrol versi populer yang banyak digunakan dalam pengembangan perangkat lunak Agile.
* Git memungkinkan tim untuk melacak perubahan kode, berkolaborasi secara efektif, dan kembali ke versi sebelumnya jika diperlukan.

**Git Commit:**

* Commit adalah snapshot dari perubahan kode yang disimpan di repositori Git.
* Commit memungkinkan tim untuk melacak kemajuan, menandai tonggak sejarah, dan berkolaborasi secara efektif.
* Untuk melakukan commit di Git, ikuti langkah-langkah berikut:
  1. Tambahkan file yang diubah ke staging area: git add <nama file>
  2. Ketik pesan commit yang menjelaskan perubahan: git commit -m "<pesan commit>"
  3. Upload perubahan ke repositori jarak jauh: git push origin <nama cabang>

**Manfaat Menggunakan Git untuk Agile:**

* Melacak perubahan kode dengan mudah.
* Berkolaborasi secara efektif dengan tim lain.
* Kembali ke versi kode sebelumnya jika diperlukan.
* Meningkatkan komunikasi dan transparansi dalam tim.
* Mendukung alur kerja pengembangan Agile yang berulang.

Penerapan alat yang tepat dan penggunaan Git secara efektif dapat membantu tim Agile meningkatkan kolaborasi, efisiensi, dan kualitas perangkat lunak mereka.

**Git sebagai Time Machine Perangkat Lunak**

Git memang dijuluki sebagai "time machine" untuk pengembangan perangkat lunak karena kemampuannya melacak perubahan kode dari waktu ke waktu. Mari kita lihat beberapa fitur Git yang mendukung julukan tersebut:

**Riwayat Lengkap:** Git menyimpan riwayat lengkap semua perubahan kode yang pernah dilakukan. Anda dapat melihat perubahan yang dibuat pada baris kode tertentu, kapan perubahan itu dilakukan, dan siapa yang membuatnya.

* **Kembali ke Masa Lalu:** Anda dapat dengan mudah "kembali ke masa lalu" dengan mengembalikan kode ke versi sebelumnya. Ini berguna jika Anda membuat perubahan yang menyebabkan bug dan perlu membatalkannya.
* **Cabang (Branching):** Fitur branch memungkinkan Anda membuat salinan independen dari kode Anda untuk bereksperimen dengan fitur baru atau perbaikan bug tanpa memengaruhi kode utama. Anda dapat beralih di antara cabang kapan saja dan menggabungkan perubahan kembali ke kode utama saat Anda siap.

**Perintah Git Penting:**

* **Git Push:** Perintah ini digunakan untuk "mengirim" perubahan kode lokal Anda ke repositori jarak jauh, seperti GitHub. Ini seperti menyimpan snapshot pekerjaan Anda di "cloud".
* **Git Pull:** Perintah ini digunakan untuk "mengambil" perubahan terbaru dari repositori jarak jauh dan menggabungkannya ke dalam kode lokal Anda. Ini seperti "memperbarui" pekerjaan Anda dengan perubahan yang dilakukan orang lain.
* **Git Branch:** Perintah ini digunakan untuk membuat, menampilkan, dan mengelola cabang kode Anda. Anda dapat menggunakannya untuk membuat isolasi untuk pengembangan fitur baru, perbaikan bug, atau eksperimen.

Dengan menggabungkan kemampuan melacak riwayat, kembali ke versi sebelumnya, dan bereksperimen dengan cabang, Git memungkinkan Anda mengontrol perubahan kode Anda layaknya mesin waktu. Ini membuat kolaborasi tim menjadi lebih mudah dan pengembangan perangkat lunak lebih efisien.

**Scrum: Panduan Singkat**

**Pengenalan Scrum:**

Scrum adalah kerangka kerja pengembangan perangkat lunak Agile yang populer untuk mengelola proyek dan produk yang kompleks. Scrum berfokus pada kolaborasi, adaptasi, dan penyampaian nilai secara berulang.

**Peranan dalam Scrum:**

* **Scrum Master:** Memfasilitasi tim Scrum dan memastikan proses Scrum berjalan dengan lancar.
* **Product Owner:** Mewakili suara pelanggan dan bertanggung jawab atas Product Backlog, yang merupakan daftar prioritas fitur produk.
* **Development Team:** Terdiri dari individu yang bertanggung jawab untuk merancang, mengembangkan, dan menguji produk.

**Sprint:**

Sprint adalah periode waktu singkat (biasanya 1-4 minggu) di mana tim Scrum fokus pada menyelesaikan sejumlah pekerjaan yang telah ditentukan dari Product Backlog.

**Sprint Event:**

Scrum memiliki empat acara seremonial yang menandai tonggak penting dalam sprint:

* **Sprint Planning:** Di awal sprint, tim merencanakan pekerjaan yang akan mereka selesaikan dan memperkirakan usaha yang diperlukan.
* **Daily Scrum:** Pertemuan singkat setiap hari di mana tim membahas kemajuan, hambatan, dan penyesuaian rencana sprint.
* **Sprint Review:** Di akhir sprint, tim mendemonstrasikan pekerjaan yang telah mereka selesaikan dan menerima umpan balik dari pemangku kepentingan.
* **Sprint Retrospective:** Tim merefleksikan sprint yang telah berlalu, mengidentifikasi apa yang berjalan dengan baik dan apa yang dapat ditingkatkan, dan membuat rencana untuk sprint berikutnya.

**Manfaat Scrum:**

* Meningkatkan komunikasi dan kolaborasi tim.
* Mengirimkan nilai secara berulang kepada pelanggan.
* Meningkatkan adaptasi terhadap perubahan.
* Meningkatkan kualitas produk.
* Meningkatkan motivasi dan kepuasan tim.

Scrum adalah kerangka kerja yang ampuh untuk pengembangan perangkat lunak yang kompleks. Dengan fokus pada kolaborasi, adaptasi, dan penyampaian nilai secara berulang, Scrum dapat membantu tim untuk menghasilkan produk berkualitas tinggi yang memenuhi kebutuhan pelanggan.

**Pengelolaan Sprint dalam Scrum:**

**Backlog:**

* Backlog adalah daftar prioritas item pekerjaan untuk produk, yang dikelola oleh Product Owner.
* Item backlog dapat berupa fitur baru, perbaikan bug, atau tugas lain yang diperlukan untuk menyelesaikan produk.
* Backlog harus terus diprioritaskan dan diperbarui untuk mencerminkan kebutuhan dan tujuan terbaru.

**Sprint Backlog:**

* Sprint backlog adalah subset dari backlog produk yang dipilih untuk dikerjakan selama sprint tertentu.
* Sprint backlog dipilih oleh tim Scrum selama Sprint Planning dan harus realistis dan dapat dicapai dalam jangka waktu sprint.
* Item sprint backlog diubah menjadi tugas yang lebih kecil dan lebih terdefinisi selama Daily Scrum.

**Velocity:**

* Velocity adalah metrik Scrum yang mengukur jumlah pekerjaan yang dapat diselesaikan tim dalam satu sprint.
* Velocity dihitung berdasarkan jumlah item sprint backlog yang diselesaikan selama sprint sebelumnya.
* Velocity dapat digunakan untuk memprediksi jumlah pekerjaan yang dapat diselesaikan tim dalam sprint mendatang, namun perlu diingat bahwa velocity dapat bervariasi tergantung pada kompleksitas pekerjaan dan faktor lainnya.

**Perencanaan Sprint (Sprint Planning):**

* Perencanaan Sprint adalah acara Scrum di mana tim merencanakan pekerjaan yang akan mereka selesaikan dalam sprint berikutnya.
* Selama Perencanaan Sprint, tim memilih item dari backlog produk, memperkirakan usaha yang diperlukan untuk menyelesaikannya, dan memecahnya menjadi tugas-tugas yang lebih kecil.
* Hasil dari Perencanaan Sprint adalah sprint backlog, yang merupakan komitmen tim untuk sprint mendatang.

**Daily Scrum:**

* Daily Scrum adalah pertemuan singkat dan informal yang diadakan setiap hari selama sprint.
* Tujuan dari Daily Scrum adalah untuk:
  + Membahas kemajuan yang dicapai pada sprint backlog.
  + Mengidentifikasi hambatan yang dihadapi tim.
  + Menyesuaikan rencana sprint jika diperlukan.
* Daily Scrum harus fokus pada apa yang telah dilakukan, apa yang akan dilakukan, dan apa yang menjadi hambatan.

**Sprint Review:**

* Sprint Review adalah acara Scrum di mana tim mendemonstrasikan pekerjaan yang telah mereka selesaikan selama sprint kepada pemangku kepentingan.
* Tujuan dari Sprint Review adalah untuk:
  + Menerima umpan balik dari pemangku kepentingan.
  + Mengidentifikasi potensi masalah atau risiko.
  + Memastikan bahwa produk memenuhi kebutuhan pelanggan.
* Sprint Review harus fokus pada produk yang bekerja, bukan pada presentasi.

**Sprint Retrospective:**

* Sprint Retrospective adalah acara Scrum di mana tim merefleksikan sprint yang telah berlalu.
* Tujuan dari Sprint Retrospective adalah untuk:
  + Mengidentifikasi apa yang berjalan dengan baik selama sprint.
  + Mengidentifikasi apa yang dapat ditingkatkan untuk sprint berikutnya.
  + Membuat rencana untuk menerapkan perbaikan.
* Sprint Retrospective harus fokus pada pembelajaran dan peningkatan, bukan pada menyalahkan atau mengkritik.

**Kesimpulan:**

Pengelolaan sprint yang efektif sangat penting untuk kesuksesan Scrum. Dengan menggunakan backlog, sprint backlog, velocity, dan acara Scrum seperti Sprint Planning, Daily Scrum, Sprint Review, dan Sprint Retrospective, tim dapat melacak kemajuan, mengelola perubahan, dan memastikan bahwa mereka terus memberikan nilai kepada pelanggan.

**Contoh user story**

User GoFood

1. Sebagai pelanggan GoFood, saya ingin dapat mencari dan memesan makanan dari berbagai restoran dengan mudah.

2. Sebagai pelanggan GoFood, saya ingin dapat melihat menu dan harga makanan sebelum memesan.

3. Sebagai pelanggan GoFood, saya ingin dapat memilih metode pembayaran yang saya inginkan.

4. Sebagai pelanggan GoFood, saya ingin dapat melacak status pesanan saya secara real time.

5. Sebagai pelanggan GoFood, saya ingin dapat memberikan rating dan ulasan untuk restoran dan makanan yang saya pesan.

6. Sebagai pelanggan GoFood, saya ingin dapat mendapatkan promo dan diskon menarik.

7. Sebagai pelanggan GoFood, saya ingin dapat mendapatkan layanan pelanggan yang baik jika saya mengalami masalah.

**bentuk task dan kelompokkan bentuk epic**

Epic 1: Pengalaman Pemesanan Makanan yang Mudah dan Lancar

User Story 1: Sebagai pelanggan GoFood, saya ingin dapat dengan mudah mencari dan menelusuri restoran berdasarkan nama, masakan, lokasi, dan filter relevan lainnya.

Task:

Implementasikan fungsi pencarian yang kuat yang mendukung berbagai kriteria pencarian.

Desain antarmuka yang intuitif untuk penjelajahan dan pemfilteran restoran.

Pastikan kinerja pencarian yang efisien dan responsif.

Epic 2: Pelacakan Pesanan Real-time dan Feedback

User Story 2: Sebagai pelanggan GoFood, saya ingin melacak status pesanan saya secara real-time, mulai dari persiapan hingga pengiriman.

Task:

Integrasikan data pelacakan pesanan real-time ke dalam platform.

Implementasikan antarmuka yang ramah pengguna untuk menampilkan pembaruan status pesanan.

Sediakan pemberitahuan push untuk perubahan status pesanan.

Epic 3: Dukungan Pelanggan dan Promosi yang Ditingkatkan

User Story 3: Sebagai pelanggan GoFood, saya ingin menerima promosi dan diskon personal yang disesuaikan dengan preferensi saya.

Task:

Implementasikan mesin rekomendasi untuk menyarankan promosi dan diskon yang relevan.

Manfaatkan data pengguna untuk mempersonalisasi penawaran promosi.

Komunikasikan promosi melalui berbagai saluran, termasuk pemberitahuan aplikasi, email, dan spanduk dalam aplikasi.