TUGAS 5
Arsitektur Perangkat Lunak

TuntasIn

untuk:

Salsabilaa

Dipersiapkan oleh:

Grup 01

Nama	NIM
Refki Alfarizi	13523002
Adhimas Aryo Bimo	13523052
Muh. Rusmin Nurwadin	13523068
Guntara Hambali	13523114
Ziyan Agil Nur Ramadhan	13622076

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132

2024

Daftar Isi

Daftar Gambar

1 *Style/Pattern* Arsitektur Acuan

Pemilihan *Model View Controller* untuk aplikasi TuntasIn didasarkan pada kebutuhan untuk mengembangkan aplikasi dengan cara membagi arsitekturnya menjadi 3 bagian, *model*, *view*, dan *controller.* Bagian model, merepresentasikan data dan logic dari aplikasi. Pada bagian inilah data akan diolah dan disimpan untuk kemudian ditampilkan ke bagian *view.* Pada bagian *view* inilah pengguna akan mendapatkan data yang telah diproses oleh model. Pada bagian ini juga, pengguna dapat mengirimkan "sinyal" kepada *controller* tentang apa saja hal-hal yang harus dilakukan. Kemudian, *controller* akan meneruskan sinyal tersebut ke *model* dan menginterpretasikannya sebagai perintah yang harus dilakukan *model*. Untuk aplikasi TuntasIn ini, proses pengembangan akan dibuat sedemikian rupa dengan urutan yaitu pengembangan bagian *model*, *view*, lalu *controller*.

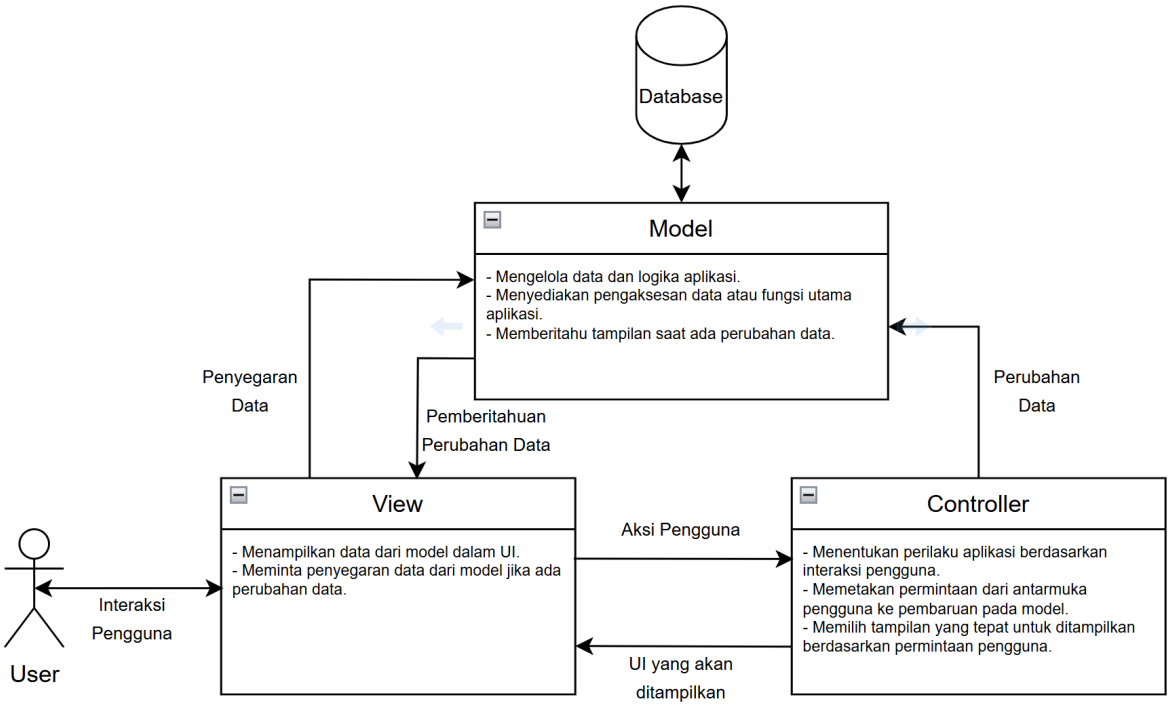
Pemilihan arsitektur MVC untuk TuntasIn memberikan berbagai keuntungan dalam hal pemeliharaan dan pemisahan komponen. Dengan MVC, logika bisnis (*model*) dipisahkan dari tampilan (*view*), sehingga memudahkan pengembangan dan pemeliharaan. Pengembang dapat memperbarui atau memperbaiki logika manajemen tugas atau mengubah tampilan aplikasi tanpa secara langsung memengaruhi komponen lainnya. Hal ini membuat proses pengembangan lebih efisien dan terstruktur. Arsitektur ini juga sangat cocok untuk aplikasi desktop dengan antarmuka pengguna yang sering diperbarui, seperti pada TuntasIn. Dalam aplikasi manajemen tugas, UI dinamis sangat penting, terutama dengan elemen yang sering diperbarui seperti daftar tugas, status tugas, dan pengingat tenggat waktu. MVC memungkinkan sinkronisasi instan antara *model* dan *view* melalui Controller, sehingga setiap pembaruan pada data langsung tercermin pada tampilan.

Selain itu, MVC mendukung antarmuka yang interaktif dan responsif. Ketika pengguna menambahkan tugas baru atau mengubah status tugas, Controller secara otomatis memperbarui Model dan memicu perubahan tampilan di View. Respons instan ini sangat bermanfaat untuk TuntasIn, yang membutuhkan pembaruan UI secara langsung berdasarkan interaksi pengguna. Arsitektur ini tidak hanya meningkatkan pengalaman pengguna tetapi juga memastikan bahwa antarmuka aplikasi selalu up-to-date sesuai dengan perubahan data. Dengan struktur yang terorganisir dan terbukti efektif, MVC menjadi pilihan yang tepat untuk aplikasi seperti TuntasIn, yang membutuhkan interaksi konstan antara pengguna dan aplikasi. MVC juga telah diakui sebagai pattern yang teruji, terutama untuk aplikasi yang memerlukan pembaruan UI secara real-time, sehingga mendukung kebutuhan interaktif dan tanggap dari aplikasi ini.

Pada tahap pertama pengembangan, akan dibuat bagian *model*. Bagian model bertanggung jawab untuk menyimpan dan mengelola data. Sebagai contoh, bagian *model* inilah yang akan bertanggung jawab terhadap perubahan data ketika terjadi pembuatan tugas baru, penghapusan tugas, pengubahan tugas, hingga penentuan waktu untuk tenggat notifikasi.

Pada tahap selanjutnya akan dibuat bagian *view*. Bagian *view* ini akan bertanggung jawab untuk menampilkan data-data yang telah diolah oleh bagian model kepada pengguna. Sebagai contoh, aplikasi akan menampilkan tugas beserta detail-detailnya. Lalu, pengguna juga dapat melihat file-file yang dilampirkan. Selain itu, bagian *view* juga akan menampilkan notifikasi mengenai tugas-tugas yang akan mendekati tenggat. Setelah itu, akan dibuat. Selain menampilkan data, bagian *view* juga bertanggung jawab untuk mengirim "sinyal perintah" hal-hal yang ingin dilakukan. Contohnya adalah bagian form untuk menambah, mengedit, dan menghapus tugas. Lalu ada juga bagian untuk mengedit tenggat waktu yang ingin dicapai.

Pada tahap terakhir, dibuatlah bagian *controller* untuk menghubungkan bagian *model* dan *view*. Bagian *controller* ini akan menjadi perantara ketika misalnya bagian *view* mengirim "sinyal perintah" (submit form, klik tombol delete, dll) ke bagian *model* untuk kemudian diolah dan disimpan. Selain itu, ketika proses pengolahan telah selesai, bagian controller akan mengirimkan lagi data dari bagian *model* ke bagian *view.*

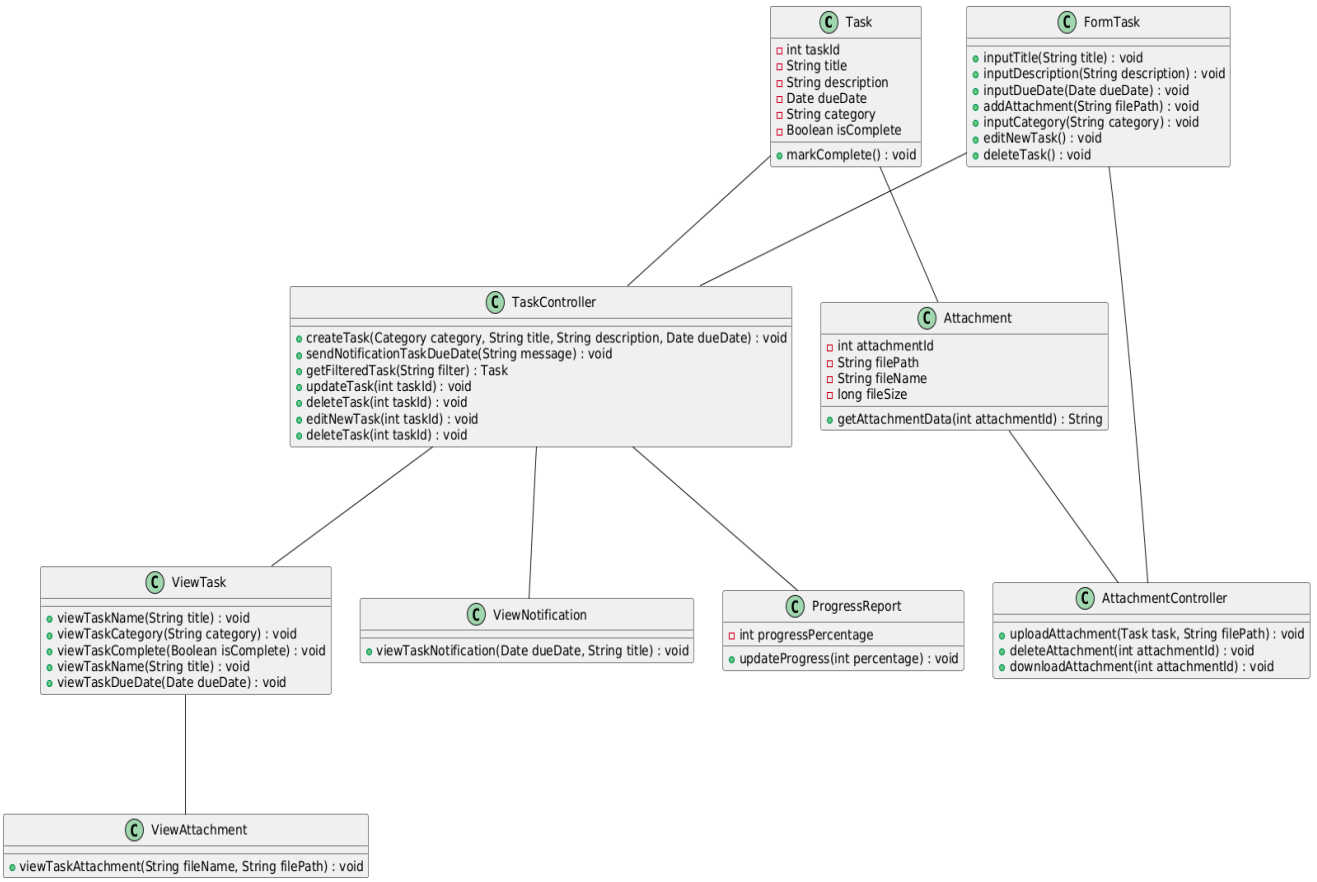


Gambar 1. *MVC* TuntasIn

2 Model Arsitektur Perangkat Lunak

2.1 Logical View

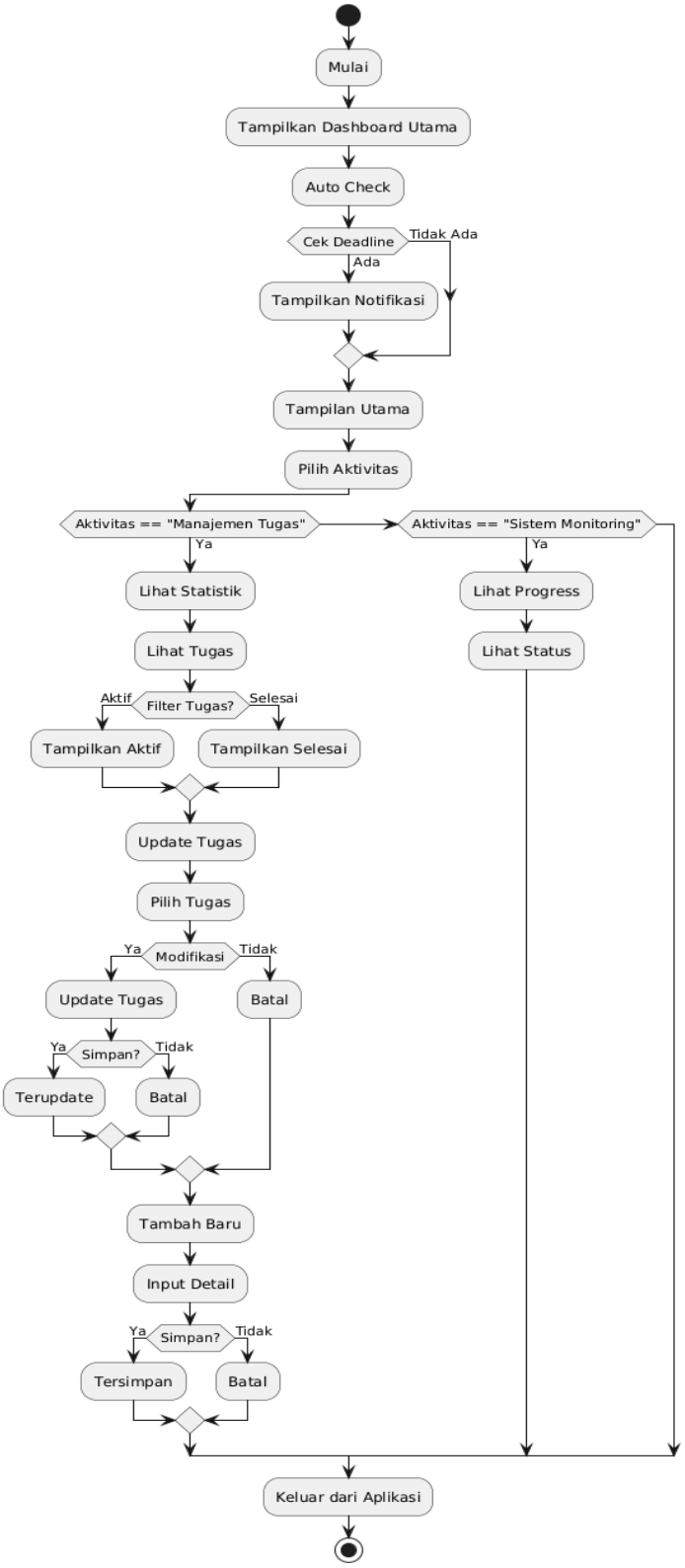
Logical View diperlukan untuk menggambarkan struktur dasar aplikasi, termasuk kelas dan relasi, sehingga pengembang memahami abstraksi utama yang membentuk sistem, misalnya melalui class diagram.



Gambar 2. *Class Diagram* Aplikasi

2.2 Process View

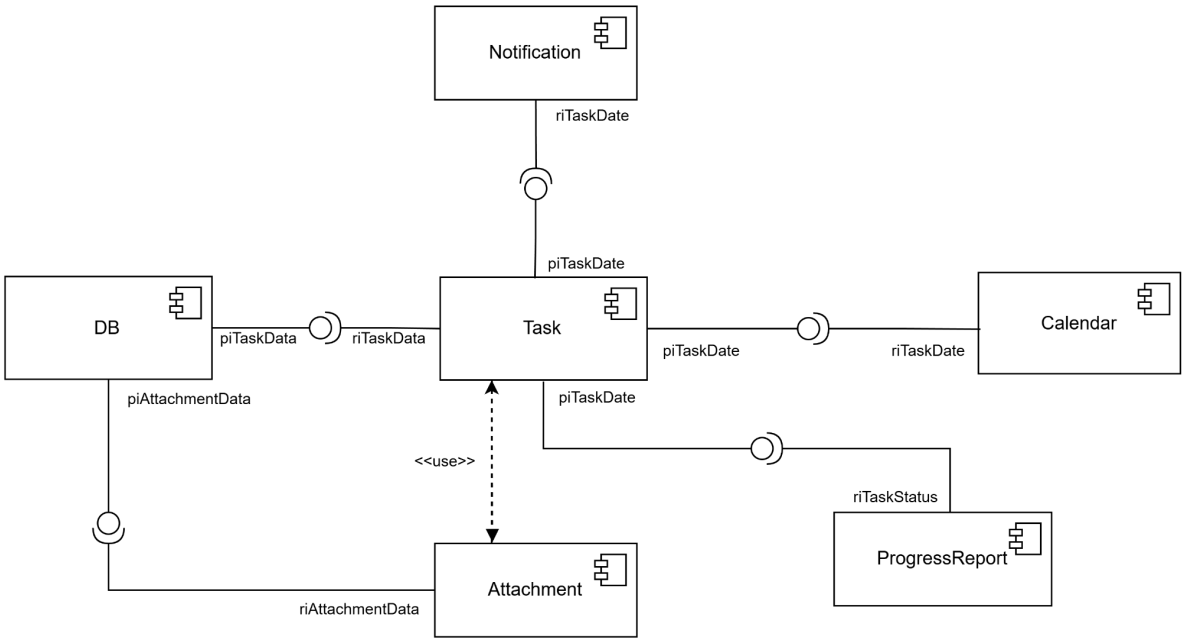
Process View membantu memetakan alur kerja dan interaksi antar-komponen selama runtime, seperti dalam mengelola tugas secara berurutan, yang divisualisasikan dengan activity diagram.



Gambar 3. *Activity Diagram* aplikasi

2.3 Development View

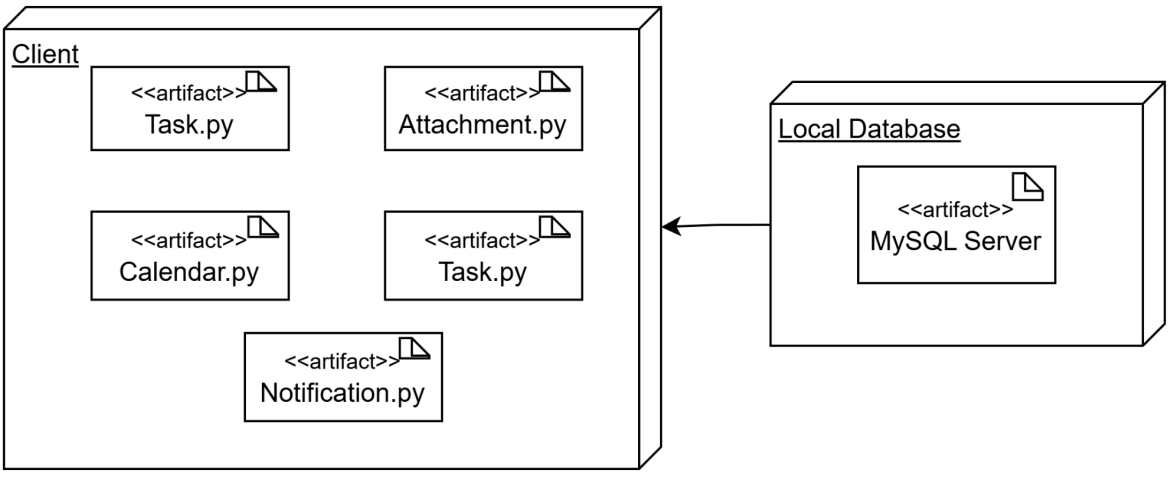
Development View (wajib) penting untuk menunjukkan bagaimana modul dan paket dikelompokkan, sehingga memudahkan pengembang dalam organisasi kode dan pembagian tugas, biasanya menggunakan component atau package diagram.



Gambar 4. *Component Diagram* Aplikasi

2.4 Physical View

Physical View adalah cara untuk menggambarkan bagaimana sebuah sistem perangkat lunak diimplementasikan dan dijalankan di atas infrastruktur fisik, seperti server dan jaringan.



Gambar 5. *Deployment Diagram* Aplikasi