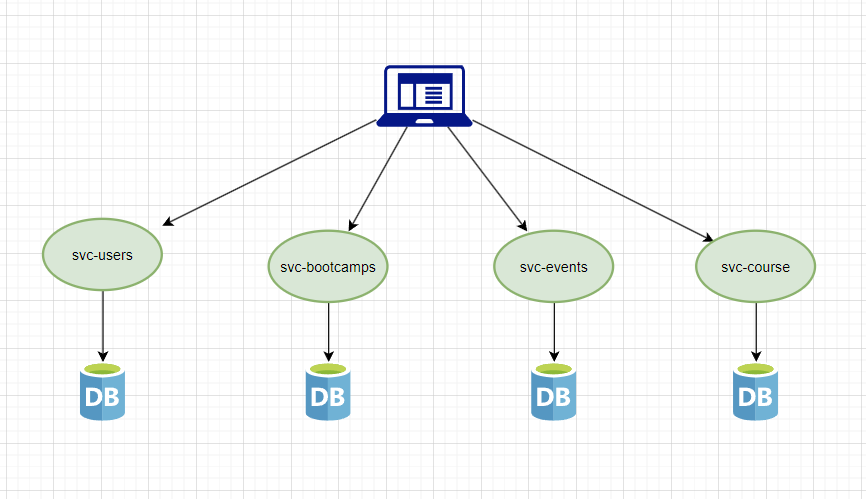
1. service:



* User Service: Menangani fitur Register dan Login.
* Bootcamp Service: Bertanggung jawab atas fitur Akses Bootcamp, Register Bootcamp, dan Logic test otomatis.
* Event Service: Mengurus fitur Akses Event, Beli Event, dan Lihat History Event.
* Online Course Service: Menangani fitur Akses Online Course, Beli Online Course, dan Lihat History Online Course.

Alasan dibuat menjadi microservice karena memisahkan setiap layanan yang bertanggung jawab atas fitur tertentu, memungkinkan tim pengembangan untuk bekerja secara independen pada setiap layanan tanpa terjadi konflik dan Jika salah satu layanan mengalami masalah, layanan lainnya dapat tetap berjalan normal, mengurangi dampak pada keseluruhan sistem. Dengan layanan yang terpisah, beberapa tim dapat bekerja secara bersamaan pada berbagai fitur tanpa mengganggu satu sama lain.

2. Bahasa Pemrograman: Golang atau Node.js karena keduanya memiliki dukungan yang baik untuk pengembangan microservices dan berbagai kerangka kerja yang cocok dengan arsitektur dan dalam mendukung concurency.  
Database yang digunakan :

* User Service: Menggunakan MySQL atau PostgreSql untuk menyimpan data pengguna seperti informasi profil, informasi login, dan data terkait pengguna.
* Bootcamp Service dan Event Service: Menggunakan MySQL atau PostgreSQL karena Informasi terkait Bootcamp dan Event dapat diatur dengan baik dalam struktur tabel yang terorganisir dan dapat diakses dengan mudah melalui SQL queries.
* Online Course Service: Basis data bisa menggunakan MongoDB untuk fleksibilitas dalam mengelola data non structural

Pendukung

* Payment Service dengan integrasi dengan gateway pembayaran seperti Xendit atau PayPal.
* MessageBroker: RabbitMQ untuk mengatur komunikasi antar layanan.
* Containerization: Docker untuk mengemas setiap layanan dalam kontainer.
* Orchestration: Kubernetes untuk mengelola dan mengorkestrasikan kontainer di lingkungan produksi.
* Cloud Platform: AWS atau Google Cloud Platform untuk menyediakan infrastruktur cloud.

3. Langkah untuk memulai proses migrasi dari monolitik ke microservice

* Menganalisa aplikasi terlebih dahulu seperti apa saja fitur-fitur yang dapat dipecah dan bisa independent dan menentukan interaksi antar layanan
* Membuat rancangan desain arsitekstur microservice
* Melakukan pemisahan layanan secara bertahap dengan memilih beberapa fitur yang enhancementnya atau pengembangannya tidak banyak atau tidak ada pengembangan di fitur itu
* Implementasi service dengan techstack yang direkomendasikan
* Melakukan testing setiap service dan komunikasi di antar service
* Melakukan migrasi data dari system monolitik ke penyimpanan data yang di rekomendasikan
* Melakukan uji integrasi menyeluruh untuk memastikan semua layanan berfungsi secara bersamaan dan sesuai harapan.
* Pemantauan dan perawatan setelah direlease

4. resource (orang) untuk melakukan migrasi dari monolitik ke microservice tergantung dari pada kompleksitas aplikasi dan pengalaman tim untuk bisa melakukan penyesuaian dengan techstack atau kebutuhan yang direkomendasikan. Sebagai perkiraan kasar, untuk migrasi aplikasi ini, mungkin diperlukan tim dengan sekitar 5-10 anggota, termasuk arsitek, pengembang, tester. Lama waktu migrasi juga bervariasi tergantung pada ukuran dan kompleksitas aplikasi serta sumber daya yang dialokasikan untuk proyek. Untuk proyek semacam ini, estimasi waktu migrasi bisa berkisar antara beberapa bulan hingga setahun.