Multi – tenancy
- Tema 10 -

În SaaS, furnizorul de servicii cloud (CSP) furnizează aplicațiile ca un serviciu, așadar clientul nu poate monitoriza sau controla ceea ce stă la baza infrastructurii. În acest caz, prin multi-tenancy se înțelege că doi sau mai mulți clienți utilizează același serviciu sau aplicație furnizate de CSP, indiferent de resursele existente.

Single tenant se referă la faptul că o instanță software și infrastructura acesteia poate fi folosită de un singur client, fiecare client având propria sa bază de date și instanță de software. Spre deosebire de acest concept, multi – tenancy reprezintă arhitectura software în care una sau mai multe instanțe de software pot fi create și executate pe lângă software-ul principal. Așadar, aceasta permite ca mai mulți utilizatori să lucreze într-un mediu software în același timp, fiecare având propria interfață, resurse și servicii, iar datele fiecărui chiriaș sunt isolate și sunt invizibile pentru ceilalți utilizatori.

Un bun exemplu pentru conceptul multi tenancy poate fi reprezentat de un bloc de apartamente. Blocul are un administrator care se ocupă de securitatea de la intrare, de electricitate, apă, dar și alte facilități. Astfel, facilitățiile oferite pot fi comparate cu soft-ul de bază ce este comun pentru mai mulți utilizatori, iar administratorul poate reprezenta furnizorul care ajută la menținerea și updatarea facilităților.

Printre beneficiile acestei arhitecturi, se numără: costurile reduse, infrastructura comună, mentenanța.

Costurile reduse față de single tenancy se datorează faptului că noii utilizatori au acces la același software de bază, intrastructurii comune. Nefiind nevoie de întreținerea infrastructurii sau a soft-ului, companiile își pot concentra atenția asupra taskurilor zilnice.

Mentenanța este considerată un beneficiu deoarece clienții nu trebuie să plătească taxe pentru aducerea la zi a software-ului, furnizorii introducând noile funcții și actualizări.

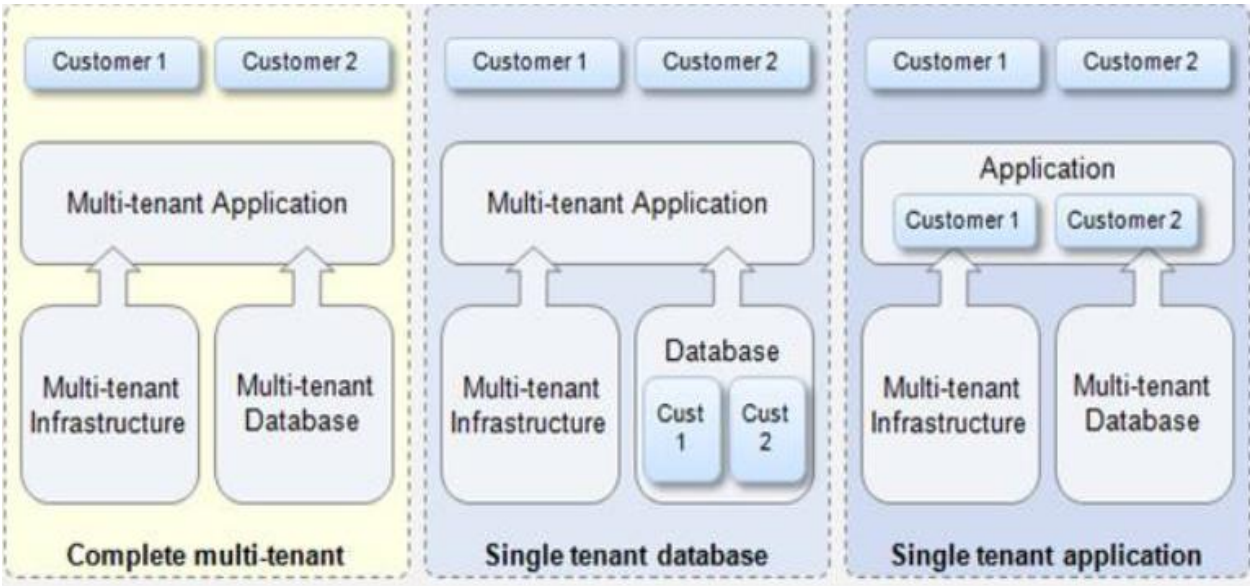
Soluțiile multi – tenant sunt construite pentru a putea fi configurate în funcție de cerințele fiecărui client, în funcție de modul în care trebuie să funcționeze aplicația.

Acest concept se bazează pe trei niveluri: data center layer, infrastructure layer și application layer. Fiecare strat este orientat către dezvoltarea de aplicații SaaS scalabile și rentabile.

La nivelul infractructurii, multi – tenancy este reprezentată de stive software, fiecare stivă fiind dedicată unui anumit client, iar disponibilitatea ridicată poate fi o opțiune pentru fiecare utilizator. Acest nivel, spre deosebire de data center layer, nu este costisitor, deoarece stivele sunt distribuite în funcție de conturile actuale ale clienților. Astfel, pe baza utilizării effective a serviciilor, cerințele hardware pot crește.

Application layer al multi tenancy necesită implementări arhitecturale atât la nivelul software-ului, cât și la nivelul infrastructurii. Modificările sunt necesare pentru o arhitectură software existente, chiar și modele multi – tenancy la nivelul aplicației. În acest caz, aplicațiile multi tenancy necesită pe lângă metodele de aplicare și tabele de baze de date pentru a accesa și a stoca date pentru diferite conturi, ceea ce poate compromite securitatea.Pentru aplicații WEB simple, application layer este o bună soluție deoarece un developer poate scrie codul mai repede și scalabil. Printre dezavantajele folosirii acestui strat, se numără și arhitectura complexă și implementarea.

Data center layer este nivelul care promite un nivel înalt de securitate dacă este bine implementat. În cele mai multe cazuri, reprezintă un serviciu prin care organizațiile pot închiria spațiu pentru stocarea datelor.



În concluzie, o arhitectură multi – tenant oferă beneficii pe termen lung furnizorilor de aplicații SaaS (Software as a Service) atât în mentenanța produsului, în development, cât și în investiții.

Nu este tocmai ușor de implementat deoarece implică anumite provocări ce trebuie identificate și castigate pe termen lung. Printre aceste provocări este întâlnită și sarcina crescută a eforturilor de dezvoltare și confidențialitate a datelor. Ele pot fi deăășite prin identificarea datelor, crearea unei arhitecturi de baze de date și proiectarea unei arhitecturi complexe a aplicațiilor.

Sebastian BUDEA
Ingineria Sistemelor
Anul IV, Grupa 1.2