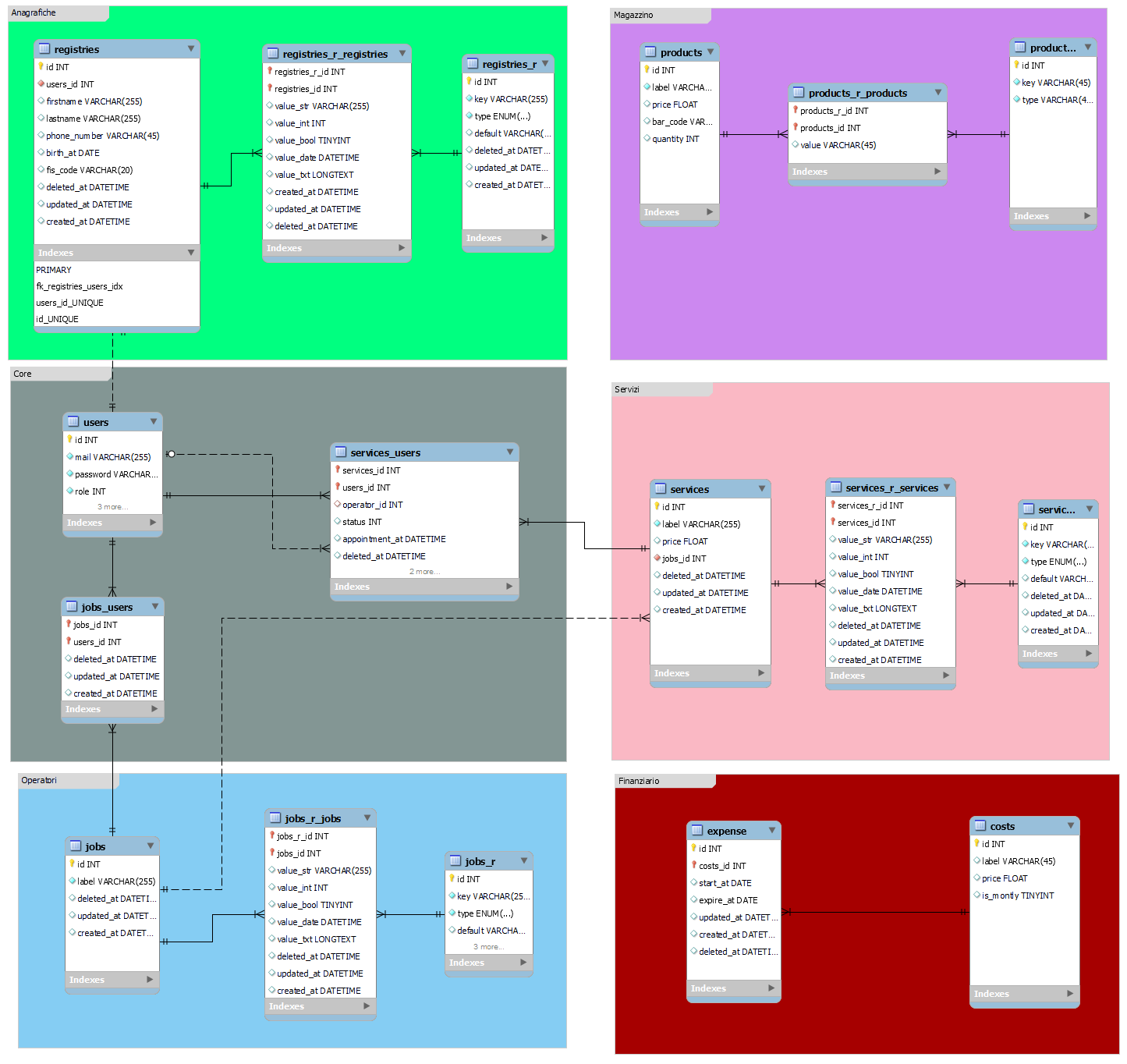
Implementazione Database



Introduzione:

Sulla base di quanto emerso nella riunione del 25/11/2022 abbiamo iniziato ad elaborare una struttura del database che rispetti l’idea di modularità che ci siamo imposti e che ci permetta di essere flessibili e agili nell’implementazione di nuovi moduli e che sia flessibile nell’adeguarsi alle diverse esigenze delle aziende che andranno ad usufruire del prodotto.

Nell’analisi seguente andremo ad escludere il modulo magazzino e quello finanziario che necessitano di ulteriori approfondimenti e analisi.

Core:

Come evidenziato la struttura centrale di Futura si basa sulla possibilità di erogare servizi di qualsiasi genere a domicilio con una marcata attenzione all’effettiva esecuzione della prestazione e il monitoraggio di queste in tempo reale. Da qui abbiamo stabilito che il centro della base dati sono gli:

* **Utenti** (Amministratori - Operatori - Clienti) -> **Users**
* **I Servizi** (Lista delle effettive offerte dell’azienda) -> **Services**
* **Le prestazioni** (Quando, dove, chi effettuerà il servizio richiesto che sarà monitorato fino all’avvenuta prestazione) -> **services\_users**

Intorno a questi dati saranno poi generati moduli per i report finanziari, gestione del magazzino etc….

Nelle premesse generali ritengo sia necessario la presenza di campi data di creazione e di aggiornamento per la maggior parte dei record se non tutti più la presenza di un campo con la data di eliminazione (deleted\_at) per poter effettuare un soft delete dei record

(Es. eliminare un servizio potrebbe comportare la perdita dei dati finanziari, in questo modo invece i servizi eliminati non saranno visibili agli utenti ma presenti in database per poter essere utilizzati nelle elaborazioni future)

Per una corretta gestione della tassonomia e dei nomi associati alle variabili propongo il suffisso \_at per le date (birth\_at, created\_at, etc….) e che i valori booleani abbiano sempre il prefisso is\_ (is\_completed, is\_verified, etc….)

Per far si che il prodotto sia customizzabile secondo le esigenze dell’azienda come proposto in riunione affiancheremo, ai campi base di una determinata entità (Es. Anagrafiche, Servizi) che saranno composti da una singola riga con i campi scelti da noi e necessari al funzionamento, una tabella con stesso nome ma suffisso \_r e con relazione molti a molti in cui sarà possibile assegnare un nome al campo e scegliere il tipo di dato. Nella tabella pivot sarà invece presente il valore associato ad ogni record e il campo riempito sarà determinato dall’attributo type selezionato

Es:

Tabella regisries\_r:

{

    "Id": 3,

    "Key": "description",

    "Type": "txt"

}

Tabella regisries\_r\_registries:

{

    "Regisries\_r\_id": 3,

    "registries\_id": 12,

    "Value\_str": null,

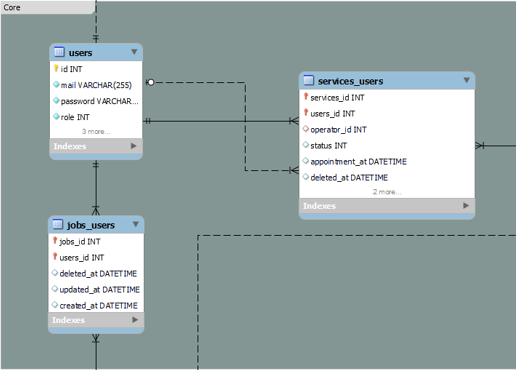
    "Value\_int": null,

    "Value\_txt": "operatore nell ambito sanitario da più di 10 anni con esperienza e campacità comunicative……"

}

Spero che l’idea di tipizzare il campo riempito non sia un’eccessiva complicazione della gestione dei campi custom generati dall’amministratore. Nel caso possiamo lasciare un solo campo value di tipo stringa che verrà convertito nel tipo desiderato dal codice stesso.

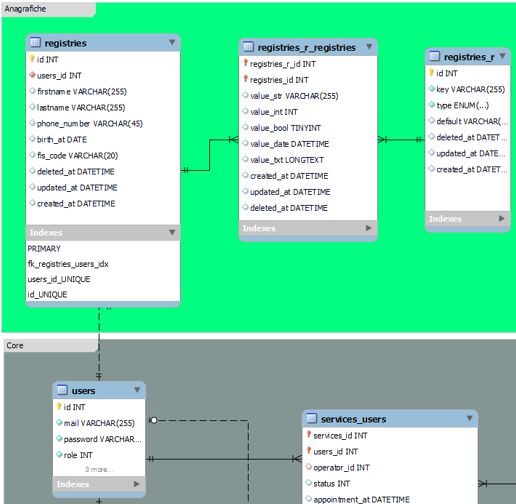
Users



La tabella Users è ridotta all’osso, possiede semplicemente i campi di autenticazione:

mail e password e il ruolo all’interno del sistema espresso con un intero (1 super admin, 2 amministratore, 3 operatore, 4 cliente etc….)

Registries



E’ la tabella anagrafiche dell’utente ed è divisa in due parti, una tabella con righe singole:

registries dove troveremo i campi anagrafici fissi per ogni installazione del prodotto che riteniamo più comuni e tutti nullabili ad eccezione del campo user\_id che dovrà essere obbligatoriamente unique.

L’altra parte è la tabella repeting ovvero registries\_r in cui ogni azenda andrà ad inserire i campi anagrafici aggiuntivi a quelli stabiliti da noi che ha intenzione di raccogliere.

Il json di ritorno per le anagrafiche potrebbe essere simile a questo.

{

    "user\_id": 3,

    "Mail": "prova@mail.it",

    "Role": 2,

    "Registries": {

        "Firstname": "giovanni",

        "Lastname": "rossi",

        "Fs\_code": "……..",

……………,

        "Registries\_r": {

            "Descrizione": "esempio descrizione.......",

            "key\_2": "value\_2"

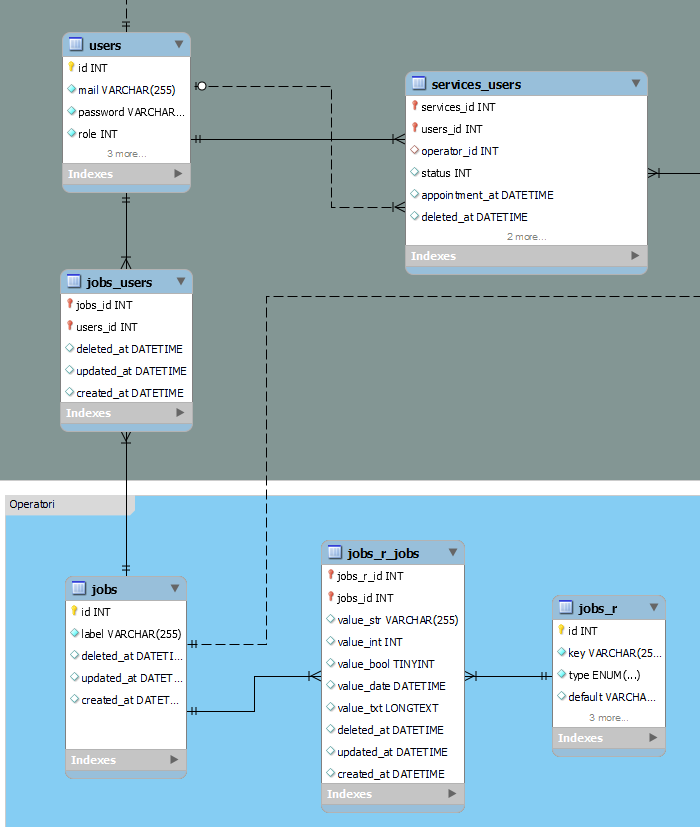
        }

    }

}

Dell’ogetto registries\_r il campo key sarà preso dalla tabella registries\_r mentre il campo value sarà preso dalla tabella pivot registries\_r\_registries.

Jobs



Con la stessa logica costruiamo la tabella jobs in cui andremo ad inserire le mansioni ricoperte da ogni operatore (es. infermiere, fisioterapista,……),

per due motivi ho pensato di stabilire la relazione operatore (tabella users) e mansione (jobs) come molti a molti.

* per tenere più pulita possibile la tabella users che così manterrà solo una funzione di log-in e registrazione.
* perché alcune strutture potrebbero pensare che un loro operatore possa compiere più mansioni.

La logica con cui è stata costruita è la medesima della tabella registries con la tabella jobs che conterrà solamente i campi stabiliti da noi. Per ora solo label ovvero il nome del job, vi chiedo quindi di suggerire se vi sono altri campi che abbiamo necessità di raccogliere per la tabella jobs.

analogamente a registries nella tabella jobs\_r troviamo i nomi dei campi aggiuntivi che l’azienda ha intenzione di raccogliere e in jobs\_r\_jobs invece il valore associato a quel campo.

{

    "user\_id": 3,

    "Mail": "prova@mail.it",

    "Role": 2,

    "Registries": {

        …….

……

    },

    "Jobs": [

        {

            "Id": 7,

            "label": "infermiere",

            "jobs\_r": {

                "label\_key\_1": "value\_key\_1",

                "label\_key\_2": "value\_key\_2"

            }

        },

        {

            "label": "fisioterapista",

            "jobs\_r": {

                "label\_key\_1": "value\_key\_1",

                "label\_key\_2": "value\_key\_2"

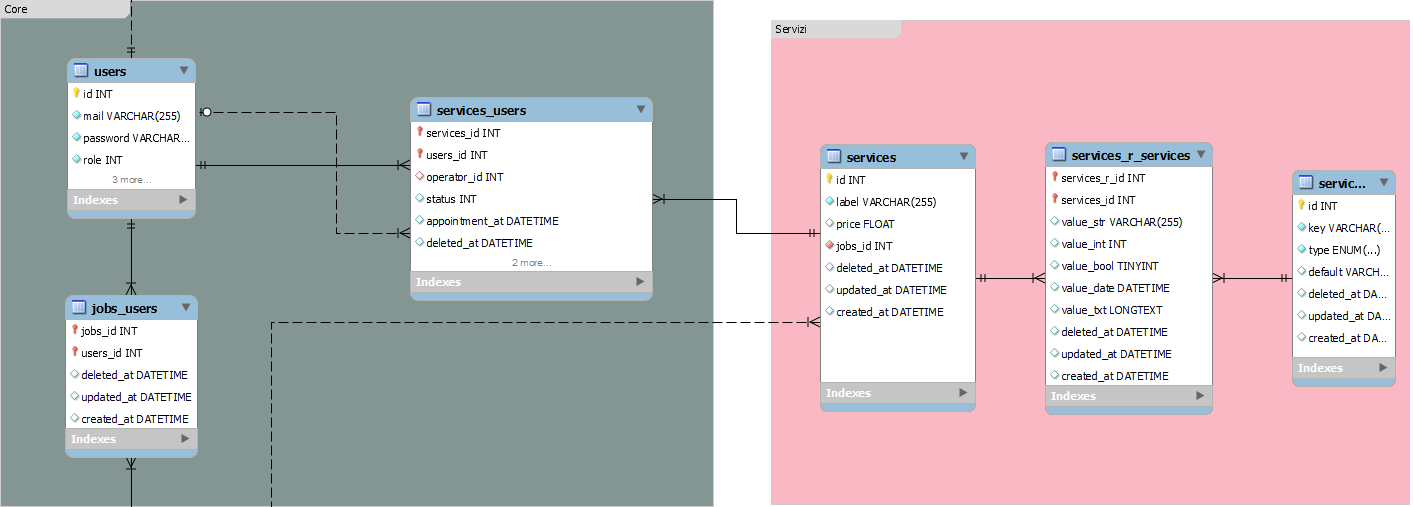
            }

        }

    ]

}

Servizi



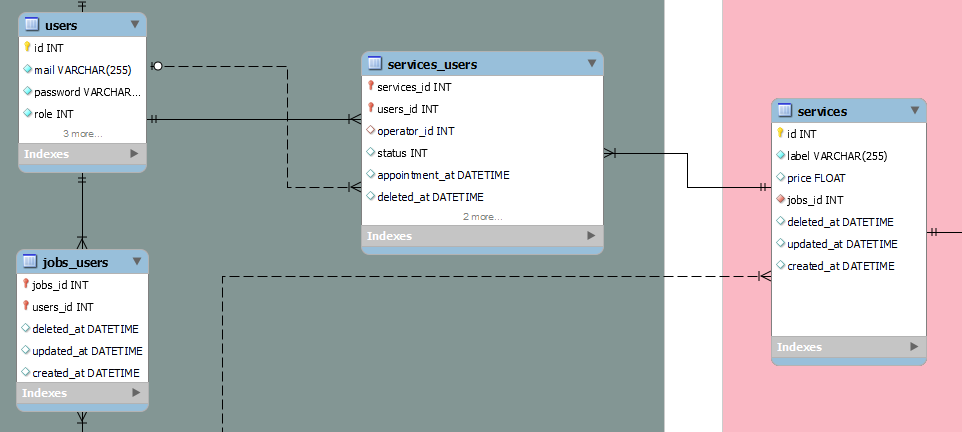
La tabella services conterrà la lista dei servizi messi a disposizione dall’azienda con associato il costo e il jobs\_id che ci permetterà di filtrare gli operatori che possono svolgere questo servizio.

I dati aggiuntivi necessari a descrivere questo servizio scelti dagli amministratori dell’azienda si troveranno in services\_r e services\_r\_services proprio come accade per i dati anagrafici e i jobs.

Anche in questo caso chiedo di ragionare su ulteriori campi oblligatori che andremo ad inserire nella tabella services con particolare attenzione alla questione delle fee (percentuali da dare all’operatore)

Su richiesta di antonio la percentuale dovrebbe essere fissa per il servizio ma che può variare anche tra un operatore e l’altro. Quindi riflettiamo su come gestirla al meglio.

Prestazioni



La tabella services\_users rappresenta di fatto una prestazione da svolgere o svolta.

La richiesta di un cliente per una servizio genera una riga in questa tabella al quale dovrà essere associato un operator\_id, ovvero un id nella tabella user associato all’operatore che svolgerà la prestazione.

Con il campo status andremo a riportare lo stato della richiesta tramite un intero, da presa in carico a assegnata all’operatore a completata.

Appointment\_at indicherà quando verrà fatta la prestazione

Propongo di riportare qui anche il prezzo a cui è stata svolta la prestazione ai fini di poter poi effettuare futuri report finanziari.

In oltre chiedo una mano su come gestire la parte relativa agli indirizzi in cui avverrà la prestazione etc….

Visto che sono sicuro che parte di queste problematiche sono state già affrontate nella prima versione dell’app.