ŠOLSKI CENTER VELENJE

**VIŠJA STROKOVNA ŠOLA**

# RAZVOJ SPLETNIH REŠITEV- HELPDESK

Deen Memić

Velenje, Januar 2025

# 1. Predstavitev projekta

## Kratek opis problema in rešitve.

V organizacijah se zahteve za podporo uporabnikom pogosto obravnavajo nepregledno: komunikacija poteka prek e-pošte, telefona ali različnih orodij, kar otežuje sledenje zahtevkom, nadzor nad odzivnimi časi, praktično reševanje problemov uporabnikov.

Moja spletna rešitev za to je sistem, ki mu pogosto v praksi pravimo »helpdesk«. Ustvaril sem preprosto a učinkovito različico, ki rešuje pogoste zgornje navedene težave uporabnikov ter seveda tudi agentom, ki te probleme rešujejo.

## Kje in komu je aplikacija namenjena (ciljna publika)

Helpdesk sistemi so ponavadi zasnovani tako, da ga lahko uporabljajo uporabniki in agenti. V smislu, da uporabniki pošljejo preko njega težavo, agent to prejme in jim odgovori nazaj.

Moja različica se pa bolj fokusira na strani agenta, saj morajo agenti ponavadi praviloma beležiti vsak zahtevek in prijavljeno težavo, ki jo prejmejo od strank preko telefona, maila, sodelavcev, nadrejenih itd. Zato potrebujejo nek »interni helpdesk« sistem, ki jim bo omogočal te napake spremljati, beležiti in imeti jasno evidenco o tem kaj je počel.

# 2. Načrtovanje aplikacije

## Načrt spletne strani

1) Namen

- Uporabniški vmesnik za upravljanje helpdesk ticketov: pregled, ustvarjanje, spremljanje SLA.

- Ciljna publika: support agenti, vodje oddelkov, skrbniki sistema.

2) Navigacija (glavna)

- Top bar ali levo navigacijsko polje:

 - Tickets

  - All Tickets

  - Create Ticket

  - SLA Monitor

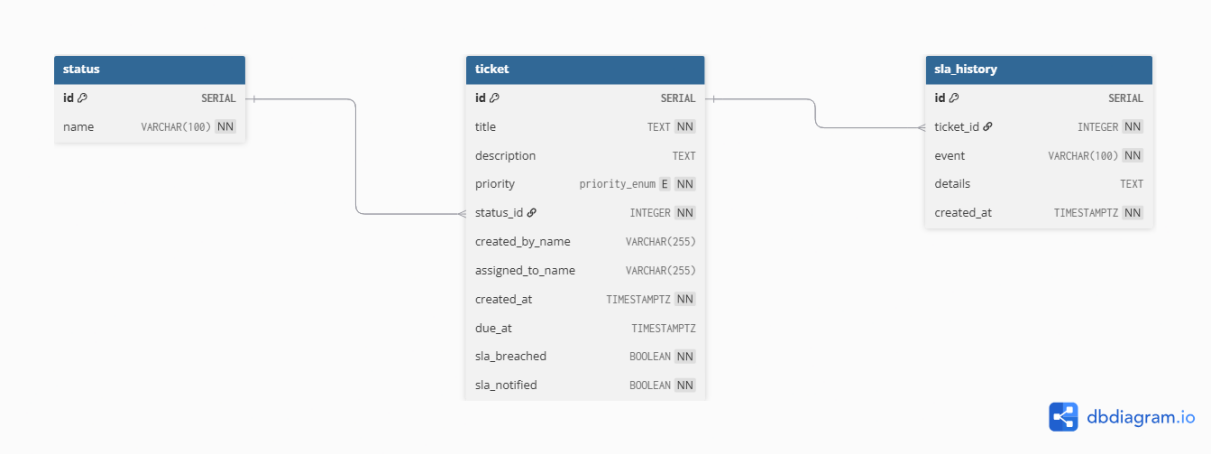
3) Strani in vsebine

- Tickets (list)
  - Tabela: ID | Title | Priority (badge) | Status (chip) | Actions.
  - Paginacija / infinite scroll.
- Ticket details
  - Header: title, id, priority badge, status dropdown.
  - Meta: createdAt, SLA status.
  - Actions: change description, change status, close ticket.
  - Right column: related tickets, attachments, SLA history.
- Create Ticket (form)
  - Polja: title, description (rich text), priority
  - Validacija in potrditev.
- SLA Monitor
  - Seznam breached ticketov, možnost masovnega obveščanja
  - CRUD

4) Oblikovna zasnova (UI/UX)

- Stil: čist, profesionalen.
- Barvna shema: nevtralne ozadja, primarna barva za CTA, rdeča za breached/high priority.
- Komponentne smernice: card, table, badge, modal, toast, form controls.
- Kontrast in tipografija za berljivost, dosledne ikone.
- Fokus na hitro izvedljive akcije in minimalno število klikov.

# Podatkovni model




**Entitete**

- status
  - Namen: slovar možnih stanj za ticket (npr. open, closed).
  - Ključna polja: id (PK), name.
- ticket
  - Namen: glavna entiteta, predstavlja podporno/servisno vstopnico.
  - Ključna polja: id (PK), title, description, priority (enum), status_id (FK → status), created_by_name, assigned_to_name, created_at, due_at, sla_breached (bool), sla_notified (bool).
- sla_history
  - Namen: zapis zgodovine dogodkov povezanih s SLA za posamezen ticket.
  - Ključna polja: id (PK), ticket_id (FK → ticket), event (npr. "breached", "reminder"), details, created_at.

**Relacije in kako delujejo v aplikaciji**

- status ← ticket (1:N)
  - Vsak ticket ima en status (status_id). En status se lahko uporablja za veliko ticketov. V aplikaciji se status spreminja za upravljanje workflow (npr. Open → In Progress → Closed).
- ticket ← sla_history (1:N)

- Za en ticket lahko obstaja več zapisov v sla_history. Aplikacija ob SLA dogodkih (npr. presežen rok, poslan opomnik) doda vrstico v sla_history z event, details in timestampom; hkrati nastavi polja na ticketu (sla_breached, sla_notified).

# 3. Tehnološki sklad in orodja

## Tehnične zahteve:

Backend (Node.js / NestJS):

- Node.js: v18.x ali višje

- npm: v9.x ali višje

- Operacijski sistem: Windows, macOS, Linux

- RAM: minimalno 512 MB

- Disk: minimalno 1 GB za node_modules in DB

- Baza podatkov: PostgreSQL 12.x ali višje

Frontend (React):

- Node.js: v18.x ali višje

- npm: v9.x ali višje

- Brskalnik: Chrome, Firefox, Safari, Edge (zadnje 2 verziji)

- Dostop do backenda na http://localhost:3000

Razvoj:

- Visual Studio Code

- Git

## 3.2 Uporabljena orodja

---

Razvoj:

- Visual Studio Code: IDE za pisanje kode

- Git: verzioniranje

Build & Runtime:

- npm: upravljanje odvisnosti

- TypeScript: statični tip-checking

- Node.js: JavaScript runtime

- Chrome: preverjanje spletne strani

## 3.3 NPM paketi — Backend

Backend: NestJS 11.0.1 + TypeScript 5.7.3 + PostgreSQL 12+ + Prisma 7.2.0

Frontend: React 18.2.0 + TypeScript 5.7.3 + Vite 4.x + Material-UI 5.x

Database: PostgreSQL 12+

Runtime: Node.js 18+

Verzioniranje: Git

# 4. Postopek razvoja in testiranje

## Kratek opis postopka izdelave

Zbral sem zahteve: opredelil sem, katere podatke mora imeti ticket in katera SLA pravila veljajo. Oblikoval sem ER model in pripravil migracije za tabele: status, ticket, sla_history. Implementiral sem REST API za CRUD operacije nad ticketi in endpoint za pridobivanje statusov. Dodal sem background worker (cron/queue), ki redno preverja SLA pogoje in nastavlja polji sla_breached ter sla_notified. Pri vsakem SLA dogodku sem v sla_history zabeležil vnos z event, details in created_at. Razvil sem preprost uporabniški vmesnik za ustvarjanje, urejanje in pregled ticketov ter zgodovine SLA.

## Opis metodologije testiranja

Testiranje se je izvajalo ročno preko spletnega brskalnika chrome-a.

# 5. Navodila za uporabo

## Kratka in jedrnata navodila za končnega uporabnika

GitHub Copilot (using GPT-5 mini)

- Kaj aplikacija nudi:
  - Upravljanje support ticketov z razvrščanjem po prioritetah
  - Spremljanje SLA statusa za vsak (high priority) ticket.

- Kaj lahko uporabnik stori:
  - Ogleda seznam ticketov in njihovo osnovno stanje.
  - Odpre podrobnosti ticketa
  - Ustvari nov ticket
  - Uredi ticket (spremeni naslov, opis, prioriteto, status).

- Vizualni namigi:
  - Rdečo ozadje / "SLA prekršeno: DA" pomeni, da je SLA ticketa prekršen.

# 6. Zaključek in samoevalvacija

## Predlogi za nadgradnje in izboljšave v prihodnosti.

Spletno aplikacijo bi lahko nadgradili z večimi uporabniškimi funkcijami. Kot naprimer iz že samih strani agentov bi lahko dodali več lastnosti ticketov, kot recimo v katerem queue-u so, in kateri lastnik jih ima. Posledično bi lahko, oziroma bi bilo smiselno omogočati opazovanje in manipuliranje svojih ter tujih ticketov. To bi omogočalo večjo funkcionalnost in večjo povezljivostjo med agenti. Za same uporabnike, ki prijavljajo podatke bi lahko ustvarili tudi portal za le njih. Omogočili bi jim lahko prijavljanje ticketov, ter opazovanje in jasen pregled kaj vse se s tem ticketom dogaja v smislu, če ga je že kdo prevzel, če je v obdelavi ipd.

# Komentar opravljenega dela: ocena lastnega vloženega truda, težave med razvojem in ocena tržne vrednosti izdelka

V projekt sem moral vložiti malo več truda, saj mi je to bil moj prvi pravi projekt v tem okolju. Stalno sem se moral izobraževati, kar je upočasnjevalo napredek izdelave aplikacije. Naletel sem tudi na mnogo težav med razvojem, ki sem si pomagal s pomočjo spleta in umetne inteligence. Z več znanja in izkušenj bi lahko ustvaril bolj funkcionalen portal.