

# **Low-Code applikationer och deras relevans i nutid**

En praktisk undersökning med fokus på databaser

**Klass:** SYSS5 + SYSJS3

**Termin och år:** vårtermin 2023

**Författare:** Fredrik Wiman, Feridoun Emin, Teodor Fredriksson, Jonathan Bravo, Lisa Dunte

**Kursansvarig:** Nihad Subasic

I det här examensarbetet vill vi ta upp ämnet Low-Code och hur det tillämpas i förenklingen av databaser. För detta definierar vi Low-Code applikationer, databaser och databashanterare och ger en översikt över teknologier som kan användas för att skapa en Low-Code applikation. Vi tar upp relevansen av Low-Code applikationer och databaser idag och presenterar slutligen vår egenbyggda applikation för att skapa en databas.

### **Förkortningar**

Low-Code Development (LCD)

Graphical User Interface (GUI)

User Interface (UI)

Relational Database Management System(RDBMS)

Structured Query Language (SQL)

Cascading Style Sheets (CSS)

Application Programming Interface (API)

<b>1. Sammanfattning</b>	<b>2</b>
<b>2. Innehållsförteckning</b>	<b>3</b>
<b>3. Uppdragsbeskrivning/problemformulering</b>	<b>5</b>
<b>4. Examensarbetets mål</b>	<b>6</b>
<b>5. Nulägesbeskrivning</b>	<b>6</b>
LowCodes relevans - nu och i framtiden	6
<b>6. Metodbeskrivning</b>	<b>8</b>
Kommunikativa verktyg	9
Miro	9
Trello	9
Figma	9
Projektdagbok	9
Discord	9
Bootstrap	10
Övrigt	10
<b>7. Resultatredevisning</b>	<b>10</b>
Vad är Low-Code?	10
Förkunskapskrav för att förstå vår app	11
Applikationens funktioner	11
Installationskrav	11
Bruksanvisning applikation	12
MySQL	12
Visual Studio Code	12
Appbeskrivning	13
Startsidan	13
Skapa databas	13
Inspektera databas	13
Sida #2	14
Gå tillbaka till startsidan	14
Skapa en tabell i den valda databasen	14
Inspektera tabell	15
Sida #3	15
Gå tillbaka till föregående sida	15
Gå tillbaka till startsidan	15
Skapa en kolumn i den valda tabellen	16
<b>8. Analys och slutsatser</b>	<b>17</b>
Vad är utmaningar med databas i kombination med lowcode?	17
<b>9. Rekommendationer</b>	<b>19</b>
<b>10. Källförteckning</b>	<b>19</b>
<b>11. Bilagor</b>	<b>21</b>
Bilaga A	22
Bilaga B	23
Bilaga C	24
Bilaga D	25

Bilaga E	26
Bilaga F	27
Bilaga G	28
Bilaga H	29
Bilaga I	30
Bilaga J	31

### 3. Uppdragsbeskrivning/problemformulering

---

Allt eftersom digitaliseringen ökar så har också behovet av kunnig arbetskraft ökat inom området (Okhrimenko, 2019). För att besvara marknadens behov och täcka bristen på arbetskraft så har Low-Code stigit i popularitet det senaste decenniet (Sufi, 2023). Low-Code har blivit ett sätt att minska kunskapskraven som ställs inom marknaden och således gjort att det är enklare att ta sig in på arbetsmarknaden för nya utvecklare (Fryling, 2019). Low-Code är snabbt, återanvändbart och kostnadseffektivt skriver Tozzi (2021).

Följaktligen så tycker vår grupp det är viktigt att undersöka vissa frågor inom Low-Code, databasområde och UI design. Syftet med vårt arbete är att skapa en applikation och den grundläggande frågan vi vill ha svar på är:

❖ *Hur kan vi i vår grupp skapa ett enkelt UI för att tillåta en användare att skapa en databas med tillhörande tabeller och kolumner?*

Andra frågor som vi tycker tillkommer när man pratar om Low-Code Development (LCD), databashantering och design av applikationer och som vi kommer besvara är:

❖ *Konceptet LCD*

➤ *Vad är Low-Code?*

➤ *Vilken relevans har Low-Code idag och för framtiden?*

❖ *Varför förenkla en databas?*

➤ *Vad en en relational database?*

➤ *Vad är vikten av relational databases idag och i framtiden?*

➤ *Varför förenkla skapandet av en databas med LCD?*

➤ *Vad är utmaningar med databas i kombination med lowcode?*

❖ *Hur kan vi skapa en enkel UI?*

➤ *Vilka koncept har vi fokuserat på för att skapa en enkel UI?*

➤ *Vilka verktyg använde vi för att skapa UI:n?*

## 4. Examensarbetets mål

---

Vårt syfte är att med hjälp av NodeJS, MySQL och ReactTS utforska hur dessa ramverk tillsammans kan skapa en applikation som utgår ifrån ett Low-Code-koncept.

Målet är att skapa en applikation med ett enkelt användargränssnitt som tillåter en användare att skapa en databas med tillhörande tabell och kolumner.

## 5. Nulägesbeskrivning

---

### **Vilken relevans har Low-Code idag och för framtiden?**

Enligt Sanchis et al. (2019) är Low-Code för många företag svaret på ökad komplexitet i företagets interna processer, marknadens snabbt rörliga och flexibla karaktär och utvecklingsmiljön ständigt växande krav. Vidare talar Sanchis et al. (2019) om Low-Code som en möjlighet att öka kapaciteten i företaget genom att avstå från traditionell mjukvaruutveckling. Han påstår att användningen av lowcode-plattformar bör betraktas som en möjlighet till motståndskraftig och effektiv utveckling (Sanchis et al., 2019).

I dagens arbetsmarknad tenderar det att vara svårt att hitta systemutvecklare. I en studie som Sanchis et al. (2019) refererar till, kom det fram att 45% av 3300 tillfrågade IT-specialister påpekade att de genom att använda Low-Code plattformar kunde minimera beroendet av specialiserade systemutvecklare. Studien också visar på att ungefär en femtedel av marknaden använder sig av Low-Code utveckling och ytterligare en femtedel av marknaden planerade att också rikta in sig mot Low-Code. Därutöver så visar studier även att kostnaden för över hälften av alla traditionellt utvecklade projekt överstiger den initiala budgeten och att nästintill en tredjedel av projekten tvingas avslutas ofärdiga (Sanchis, 2019, s. 8 ). Sanchis et al. (2019, s. 8) uttrycker vidare att populariteten av Low-Code förväntas stiga.

Beranic et al. (2020) talar även om en prognos där det antas att 2024 kommer mer än 65 % av alla applikationer att genereras med hjälp av Low-Code eller No-Code lösning.

Ovan referat pekar på en framtid där Low-Code kommer ha allt större betydelse för många företag, oavsett om du kommer från en tekniskt avancerad bakgrund eller inte. Med det i beaktning anser vi att det finns starka argument för att vidare utforska och konkretisera möjligheterna med Low-Code.

## Vad är vikten av relational databases idag och i framtiden?

Enligt DB-Engines hemsida, som listar de mest populära datahanteringssystemen, anses MySQL vara det viktigaste datahanteringssystemet efter Oracle (solid IT gmbh, 2023). Uppsatsen av Kolonko (2018) påpekar redan vikten av MySQL och hänvisar till DB-Engines som en webbplats för att kartlägga populariteten för datahanteringssystem.

Rang			DBMS	Datenbankmodell	Punkte		
Mär 2023	Feb 2023	Mär 2022			Mär 2023	Feb 2023	Mär 2022
1.	1.	1.	Oracle +	Relational, Multi-Model ⓘ	1261,29	+13,77	+9,97
2.	2.	2.	MySQL +	Relational, Multi-Model ⓘ	1182,79	-12,66	-15,45
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-Model ⓘ	922,01	-7,08	-11,77
4.	4.	4.	PostgreSQL +	Relational, Multi-Model ⓘ	613,83	-2,67	-3,10
5.	5.	5.	MongoDB +	Document, Multi-Model ⓘ	458,78	+6,02	-26,88
6.	6.	6.	Redis +	Key-value, Multi-Model ⓘ	172,45	-1,39	-4,31
7.	7.	7.	IBM Db2	Relational, Multi-Model ⓘ	142,92	-0,04	-19,22
8.	8.	8.	Elasticsearch	Suchmaschine, Multi-Model ⓘ	139,07	+0,47	-20,88
9.	9.	↑ 10.	SQLite +	Relational	133,82	+1,15	+1,64
10.	10.	↓ 9.	Microsoft Access	Relational	132,06	+1,03	-3,37

(Bild 1: Lista med alla databashanterare, hämtad den 02 mars 2023 från <https://db-engines.com/de/ranking>)

Med tanke på vikten av MySQL under de senaste fem åren anser gruppen det som rimligt att utforska skapandet av en Low-Code applikation för en MySQL databas med syftet på att underlätta hanteringen.

## Varför förenkla skapandet av en databas med LCD?

I takt med att det globala samhället ställer allt mer komplexa krav i form av digitalisering kräver det också på individnivå en ökad förmåga att bidra till samhälls- och företagsstrukturer på ett teknologiskt plan. (Okhrimenko, 2019)

Bäckström (2021) nämner behovet och fördelarna av att digitalisera analoga samt fysiska arbetsplatser. De tar upp exempel på arbetsprocesser där utbyte av information mellan avdelningar inom ett företag sker via papperskopior. Flera av de analyserade arbetsprocesserna

beskrivs som tidskrävande, ineffektiva och förknippade med risk för förlust av information (Bäckström, 2021).

Arbetsplatser rapporterade att vissa analoga processer som digitaliserats hade effektiviserat arbetsflödet med upp till 90% (Bäckström, 2021).

Utifrån dessa studier och tidigare nämnda utvecklingstrender anser vi det rimligt att digitaliseringen av arbetsflöden med hjälp av Low-Code -koncept kommer fortsätta att öka. Vår avgränsning för implementering av Low-Code sker vid skapandet och uppdaterandet av databaser då det är troligt att individer ansvariga för informationshantering kommer att använda sig av Low-Code-genererade databaser utan att behöva gedigen erfarenhet inom området.

## 6. Metodbeskrivning

---

Under arbetets gång har vi använt oss av diverse tekniska och kommunikativa verktyg. Nedan nämner vi dessa verktyg och kort varför vi ansåg de nödvändiga eller användbara.

### **Tekniska verktyg**

- NodeJS med Express för API - smidigt sätt att sätta upp en server.
- MySQL för databashantering - skalbar, smidig samt tidigare erfarenhet av verktyget
- TypeScript för typsäkerhet - en metodik som efterfrågas allt mer
- React för återanvändning av frontendkomponenter - tidseffektivt
- Figma - smidigt sätt att visualisera projekt
- Bootstrap för färdig CSS - tidseffektivt
- Mob Programming - underlättar att gemensamt upptäcka kod som kan leda till buggar samt på gruppnivå snabbt bidra med kreativa och effektiva lösningar



## **Kommunikativa verktyg**

### **Miro**

Miro gav oss ett verktyg för att enklare strukturera och planera arbetet. I Miro så placerade vi ut hur vi skulle gå tillväga, ungefär som en virtuell whiteboard. Vi kunde sätta upp tider, regler för gruppen, idéer och brainstorming på intressanta frågor att försöka besvara med hjälp av studier.

### **Trello**

Trello använde vi för att följa upp det tekniska arbetet. Vi skapade olika tabbar med en backlog, en sprint log, ongoing, In Review/Test och slutligen Done. Vi flyttade runt arbetsuppgifterna efter deras status.

### **Figma**

Med Figma kunde vi planera vår frontend. Vi kunde med Figma lättare se också vad som skulle vara intressant att ha med och bilda en gemensam bild av själva programmet.

### **Projektdagbok**

Dagboken fylldes i dagligen och det hjälpte oss att lättare kunna se vart vi är i projektet, ett förtydligande till Trello där man kunde lägga in mer nyans med vilka delar vi implementerat i programmet. Dagboken fungerade även som en form av backtracking då vi kunde se vår motivering till varför vi valt en viss kodstruktur, och ta med detta i beräkningen i de fall vi valde att revidera koden.

### **Discord**

Discord var bra för att sköta all kommunikation med chatt- och röst-kanaler. Med discord så kunde vi köra “mob-programming” vilket innebär att en streamar sin skärm och skriver kod medan andra kommenterar vad som borde skrivas. Det var ett bra sätt att få in allas perspektiv och att göra alla delaktiga i kodningen. Vi hade också länkar till dom olika ovanstående delarna och kunde lättare kommunicera eventuella förhinder i form av sjukdom, förseningar osv.

Bootstrap var ett effektivt verktyg för att snabbt kunna använda befintlig CSS-kod och enbart lägga till små justeringar som passade vår layout.

### Övrigt

Under arbetets gång hade vi kontinuerlig dialog om vilken typ av information eller kunskap som behövde förstärkas. Oftast skedde efterforskningen tillsammans inom ramen för “mob-programming”, ett koncept som beskrivits i mer detalj ovan.

## 7. Resultatredovisning

---

### Vad är Low-Code?

Lefort (2019) beskriver Low-Code som en typ av utveckling där det traditionella förhållningssättet till programmering ersätts med användning av ett användargränssnitt. Enligt Lefort (2019) ligger fokus här på en modellbaserad design som gör att användare utan förkunskaper kan lösa programmatiska problem. Han säger också att många plattformar tillåter användare att göra ändringar genom att själva lägga till kod.

Törnqvist (2021) argumenterar för att det finns en skillnad mellan low-code och no-code, som ofta benämns som en avart från Low-Code. Han anser att no-code innebär att användaren ska kunna skapa och lansera programvara utan några tekniska förkunskaper över huvud taget.

Motparten low-code, genererar istället vad man kan kalla för grunden och bulken av en kodbas, där användaren förväntas ha de tekniska kunskaperna som krävs för att binda ihop kodstrukturen. Detta innebär samtidigt att användaren har större möjlighet att anpassa sin kod efter egna önskemål, till skillnad från no-code konceptet som mer eller mindre begränsar användandet till helt färdiga koncept och mallar (Törnqvist, 2021).

Utifrån ovanstående definitioner har vi valt att definiera Low-Code som ett koncept som i största grad kräver någon grad av teknisk förkunskap. Vi förstår Low-Code som ett koncept som ska underlätta och effektivisera arbetsflödet och minskar samtidigt glappet mellan kunskap och utförandet. Vi tolkar det som att konsensus verkar finnas i att det är genom ett användarvänligt gränssnitt som Low-Code tillämpas, oavsett tekniska förkunskaper.

## **Vilka koncept har vi fokuserat på för att skapa en enkel UI?**

Vi valde tidigt att utveckla ett användarvänligt och avskalat visuellt gränssnitt för appens funktionalitet. Även om Graphical User Interface (GUI) starkt förknippas med No-Code, så kan dessa koncept också implementeras inom Low-Code. Eftersom vi valde att skapa en UI och många av appens visuella komponenter naturligt utvecklades utifrån egna behov och en översyn av funktionalitet, var vi tvungna att granska grundläggande principer inom UI-design. Vi använde dessa som referenser för att förstärka en aspekt av Low-Code, nämligen effektivisering av arbetsflödet.

Några av de viktigaste UI-designkoncepten vi försökte följa var System Consistency, där genomgående design i form, färg och funktionalitet är sammansvetsad, Relevance of Content, som berör hanteringen av relevant data användaren ska exponeras för, och Error Handling med övergripande visualisering av fel och möjligheten att hantera det och gå vidare med lösningar (Guntupalli, 2008).

## **Vilka verktyg använde vi för att skapa UI:n?**

Till vår hjälp använde vi Bootstrap för att underlätta CSS-syntaxen och ge utrymme för designutveckling av appens användargränssnitt (UI). Bootstrap är känd för sin popularitet inom webbutveckling på grund av sin “rapid, responsive development that is consistent and well supported by the development and design community” skriver Gaikwad och Adkar (2019).

Dessutom valde vi React som ramverk för att bygga vårt UI. Rawat och Mahajan (2020) beskriver react som ett intelligent och dynamiskt ramverk med en kort och enkel inlärningskurva. Det motsvara våra egna erfarenheter med react.

## Vad är en relational database?

Harrington (2016, s. 4) definierar termen relationsdatabas som en samling av listor med relevant information för företaget t.ex. namn, adresser och telefonnummer. Databasen innehåller dock inte bara själva informationen utan också information om förhållandet mellan data i två eller flera olika listor (Harrington, 2016, s. 5). Leven och Loizou (2021) definiera en databas som en “organised collection of logically inter-connected *data items*”.

I avgränsning mot detta står relationsdatabashanteringssystemet (RDBMS), som är ett verktyg som använder ett relationellt frågespråk (t.ex. SQL) för att ge användaren enkel åtkomst till databasen (Melton & Simon, 1993).

## Förkunskapskrav för att förstå vår app

Kravet som ställs på användaren av vår applikation är enbart en grundläggande förståelse om databaser, allt annat gör applikationen åt användaren av programmet. Det saknas givetvis vissa funktioner som att ta bort kolumner och lägga till information i kolumnerna. Men i mån av tid så valde vi att begränsa oss.

## Applikationens funktioner

Syftet med appen var att göra en liten demo för klargöra vad konceptet Low-Code går ut på. I vårt program så kan en person kunna:

- *Skapa en databas i MySQL*
- *Lägga till tabeller i vald databas*
- *Lägga till kolumner i vald tabell*

Det som behövs för att kunna använda applikationen är följande:

## Installationskrav

För att köra applikationen finns det följande installationskrav;

- Visual Studio Code: <https://code.visualstudio.com/download>
- NodeJS: <https://nodejs.org/en/download/>

- MySQL: <https://dev.mysql.com/downloads/installer/>

## Bruksanvisning applikation

### MySQL

- Logga in på MySQL
- Navigera till fliken 'Administration'
- Klicka på fliken 'Users and Privileges'
- Klicka på knappen 'Add Account'
- Fyll i följande användaruppgifter
  - Login Name: examUser
  - Authentication Type: Standard
  - Limit Hosts Matching: Localhost
  - Password: 123123
  - Confirm Password: 123123
- Klicka på fliken 'Administrative Roles'
- Fyll i rutan 'DBA' under titeln 'Role'
- Klicka på knappen 'Apply'

Du kan nu minimera MySQL, men stäng inte ned fönstret. Om du gör det måste du starta MySQL och logga in igen.

### Visual Studio Code

- Starta Visual Studio Code
- Klona ned projektet från GitHub: <https://github.com/LisaDuenete/ExamensArbete.git>
- Klicka på fliken 'Terminal' → välj alternativet 'New Terminal'
- I terminalen skriv 'cd .\examen\src\server\'
- Sedan skriv 'node server.js' för att starta servern
- I terminalen klicka på + tecken på höger sidan
- I terminalen, skriv 'npm run start'

Applikationen ska nu starta. Din webbläsare borde starta av sig själv och ta dig till startsidan.

### Startsidan

På startsidan kommer du se vilka databaser som finns tillgängliga just nu. Till att börja med kommer du se ett antal redan befintliga databaser ([se Bilaga A](#)). Dessa databaser skapas per automatik i samband med installationen av MySQL. I denna applikation lämpar de sig väl för att demonstrera en första visuell presentation av gränssnittets tillgängliga databaser.

På startsidan har man två alternativ;

- Skapa databas
- Inspektera databas

### Skapa databas

Om man klickar på 'Create Database'-knappen får man en pop-up modal som låter användaren skriva in ett namn på en ny databas ([se Bilaga B](#)). Om man skriver in ett önskat namn (som inte redan existerar) och klickar på 'Create'-knappen skapas databasen och användaren skickas till startsidan där användaren nu kan se en uppdaterad lista av tillgängliga databaser ([se Bilaga C](#)).

Om namnet man valt för sin databas använder sig av syntax som databashanteraren inte kan läsa formateras namnet om för att passa kraven. Se nedan exempel på formatering av namn;

- 'MinTestDataBas' → 'mintestdatabas'
- 'min test databas!' → 'min\_test\_databasexclamationmark'
- test databas → 'test\_databas'

Klickar användaren på 'Cancel'-knappen vid något tillfälle skickas användaren tillbaka till startsidan utan att någon databas läggs till ([se Bilaga A](#)).

### Inspektera databas

Om man klickar på en av de tillgängliga databaserna skickas användaren till sida #2. I detta exempel har vi valt att inspektera vår nyligen skapade databas 'test\_database' ([se bilaga D](#)).

På sida #2 kan vi se hur vår databas ‘**test\_database**’ ser ut och vilka tabeller den innehåller. För tillfället är databasen tom och innehåller inga tabeller ([se Bilaga D](#)). Vi kan hålla koll på vilken databas vi för tillfället befinner oss i genom att titta på rubriken som finns mellan ‘Create Table’-knappen och ‘Back’-knappen ([se Bilaga D](#)).

På sida #2 har man tre alternativ;

- Gå tillbaka till startsidan
- Skapa en tabell i den valda databasen
- Inspektera en tabell

### Gå tillbaka till startsidan

Om man klickar på ‘Back’-knappen kommer användaren tillbaka till startsidan och är fri att på nytt välja vilken databas denne vill inspektera.

### Skapa en tabell i den valda databasen

Om man klickar på ‘Create Table’-knappen får man en pop-up modal som låter användaren skriva in ett namn på en ny tabell ([se Bilaga E](#)). Om man skriver in ett önskat namn (som inte redan existerar) och klickar på ‘Create’-knappen skapas tabellen och användaren skickas till sida #2 där användaren nu kan se en uppdaterad lista av tillgängliga tabellen för den valda databasen ([se Bilaga F](#)).

Om namnet man valt för sin tabell använder sig av syntax som databashanteraren inte kan läsa formateras namnet om för att passa kraven. Se nedan exempel på formatering av namn;

- ‘MinTestTabell’ → ‘mintesttabell’
- ‘min test tabell!’ → ‘min\_test\_tabellexclamationmark’
- ‘Min Super Test Tabell’ → ‘min\_super\_test\_tabell’

Klickar användaren på ‘Cancel’-knappen vid något tillfälle skickas användaren tillbaka till sida #2 utan att någon tabell läggs till ([se Bilaga D](#)).

Om man klickar på en av de tillgängliga tabellerna skickas användaren till sida #3. I detta exempel har vi valt att inspektera vår nyligen skapade tabell ‘**min\_super\_test\_tabell**’ ([se Bilaga G](#)).

### **Sida #3**

På sida #3 kan vi se hur vår tabell ‘**min\_super\_test\_tabell**’ ser ut och vilka kolumner den innehåller. För tillfället innehåller tabellen bara en kolumn ‘**id**’ som autogenererades när vi skapade vår första tabell, ‘**min\_super\_test\_tabell**’ ([se Bilaga G](#)). Vi kan hålla koll på vilken databas vi för tillfället befinner oss i genom att titta på rubriken som finns mellan ‘Create Column’-knappen och ‘Back’-knappen ([se Bilaga G](#)).

På sida #3 har man tre alternativ;

- Gå tillbaka till föregående sida
- Gå tillbaka till startsidan
- Skapa en kolumn i den valda tabellen

### Gå tillbaka till föregående sida

Om man klickar på ‘Back’-knappen kommer användaren tillbaka till sida #2 och är fri att på nytt välja vilken tabell denne vill inspektera, alternativt skapa en ny tabell.

### Gå tillbaka till startsidan

Om man klickar på ‘Back to main’-knappen kommer användaren tillbaka till startsidan och är fri att på nytt välja vilken databas denne vill inspektera.

### Skapa en kolumn i den valda tabellen

Om man klickar på ‘Create Column’-knappen får man en pop-up modal som låter användaren fylla i följande ([se Bilaga H](#));

- namn på kolumn<sup>1</sup>

---

<sup>1</sup> Följer samma formateringsmetodik som namngivande av databaser och tabeller.



- datatyp<sup>2</sup>
- primary key<sup>3</sup>

Om man fyller i dessa fält och trycker på ‘Create’-knappen skapas kolumnen och användaren skickas till sida #3 där användaren nu kan se en uppdaterad lista av vilka kolumner som finns lagrade i den valda tabellen ([se Bilaga I](#)).

Klickar användaren på ‘Cancel’-knappen vid något tillfälle skickas användaren tillbaka till startsidan utan att någon kolumn läggs till ([se Bilaga G](#)).

Ovan bruksanvisning finns även visuellt presenterad ([se Bilaga J](#)).

## 8. Analys och slutsatser

---

### Återblick av tillvägagångssättet

I vårt arbete har vi nu redogjort för hur vi gått tillväga för att skapa en applikation för att förenkla databasen med tanken på konceptet Low-Code och sedan även gett en förklaring på relevansen av arbetet vi gjort med stöd av studier och artiklar. I skapandet av applikationen har vi gjort det så enkelt som möjligt för att någon utomstående ska kunna använda sig av den. Förkunskaper om databaser krävs men en avsevärd del sköter själva applikationen åt användaren av den och vi hoppas att det tydliggjort konceptet Low-Code. Som ordet insinuerar så behövs det en låg förståelse av kod för att förstå hur man använder vår applikation.

Vi anser att verktygen vi använt oss av är relevanta, smidiga och att grunden vi har lagt har gjort att projektet utan större svårigheter hade kunnat skalas upp. Bootstrap var ett bekvämt val i ett så här litet projekt och det går även tänka sig att det fungerar att använda i längden då vi tycker det håller användargränssnittet stilrent och simpelt. Användningen av bootstrap gör också applikationen responsiv. Valet av MySQL motiveras med att det är skalbart, håller en bra prestanda och att det var enkelt att använda i just vårt projekt. Det är även ett system som är

---

<sup>2</sup> Datatyper är något som alltid används man lagrar data i en databas. Datatypen berättar för databashanteraren vilken typ av data den kan förvänta sig hantera. Det finns ett stort antal datatyper som passar olika bra för olika typer av data. Två exempel är textsträngar och siffror. Datatypen för textsträngar kallas ofta för ‘**varchar**’. Använder man ‘**varchar**’ behöver man även ange hur många tecken textsträngen kan innehålla totalt, ex ‘**varchar(255)**’. Datatypen för siffror mellan -2147483648 och 2147483647 kallas ofta för ‘**int**’.

<sup>3</sup> En primary key är en kolumn i en RDBMS som är utmärkande för varje enskild tabell och är en av grundstenarna till relationsbaserad databaser. En primary key är unik och kan liknas vid ett personnummer eller en bils registreringskylt. Primary keys hjälper till att identifiera unik data.

högst aktuellt på marknaden. Med React kunde vi återanvända kod vi redan skrivit och la en god grund för att lägga till ytterligare funktioner som använder samma komponenter. TypeScript garanterade en viss typ-säkerhet för projektet och NodeJS i kombination med express ramverket gjorde det enkelt att lägga upp en server som vi sedan kunde anropa på för att hämta eller lämna information till databasen.

## **Vad är utmaningar med databas i kombination med lowcode?**

Utifrån vårt eget perspektiv är vår demo av Low-Code en simpel applikation utan några till synes större risker för användare att göra fel. Vi anser det dock rimligt att nyansera potentiell problematik med applikationen ifall den skulle skalas upp ex inför en lansering riktad till en större publik.

Problem som kan uppstå med low code är att säkerheten hos Low-Code plattformen kan vara olämplig för den verksamhet man bedriver och man har ingen eller åtminstone en sämre insyn i hur source-koden är strukturerad. Ett annat problem är att utvecklare kan bli låsta till plattformen (Tozzi , 2021).

En studie som utförts av Luo et al., (2021) som syftat till att undersöka utmaningar med Low-Code instämmer i ovanstående och tillägger flera problemområden som skalbarhet, flexibilitet, hög inlärningskurva även för nybörjare, höga priser, svårt att debugga och hitta fel, komplexa problem kan fortfarande behöva traditionell kodning. Och slutligen att utgivarna av low code plattformen kan välja att stänga ner tjänsten vilket skulle kunna förorsaka betydande svårigheter för verksamheten (Luo et al., 2021).

Vår uppfattning blir följaktligen att även om Low-Code garanterar en viss säkerhet när det kommer till hur konsekvent det kan vara så betyder det inte nödvändigtvis att den inte brister inom andra säkerhetsområden. Low-Code verkar inte alltid vara lämpligt att använda inom områden som sätter en hög betoning på säkerhet när det kommer till intrång, skalbarhet, rigorös felhantering eller där behovet av traditionell kodning ofta återkommer.

Det finns enorma begränsningar och risker med vår applikation men för att återkoppla till syftet så har vårt fokus enbart varit att konstruera en enkel demonstration. Men det går att tänka sig att en mer mogen och utvecklad version av vad vi åstadkommit hade kunnat användas på olika

ställen där personer kan tänkas behöva använda sig av databaser och allt som hör därtill på ett smidigare och enklare sätt än att behöva ha en god förståelse av SQL-språket.

## **Kan vår kod anses vara No-Code?**

En tanke vi har är att vi valde att ha ett visuellt gränssnitt som kan göra att applikationen möjligen kan anses vara en No-Code applikation. Förklaringen vi har där är att just nu måste man ha en användarprofil på MySQL och vår applikation bara är en liten demo som i mån av hur lite tid vi har blev ytterst begränsad.

## **Vidareutveckling av vår applikation**

En vidareutveckling som vi ser för vår applikation utöver dom funktioner som redan finns hade varit att kunna lägga till/ta bort information i kolumner. Begränsa tillgången till specifika användare och därmed öka säkerheten. Man kan tänka sig att vår applikation kan vara en komponent i ett större program som avser att göra databashantering enklare. För att höja vårt program till en äkta Low-Code applikation skulle det vara nödvändigt att skriva ett bibliotek som skulle tillåta användaren att implementera en skräddarsydd lösning för sina egna problem, till exempel i form av ett användargränssnitt för att skriva egna SQL-queries .

## **9. Rekommendationer**

---

Efter omfattande forskning kring konceptet Low-Code och MySQL-databasens aktualitet, förespråkar gruppen att man håller ett öga på utvecklingen av Low-Code plattformar och företagen som utvecklar dem. På grund av en ökande digitalisering av världen och brist på utvecklare inser vi vikten av plattformar som stödjer snabb åtkomst till skapandet av kod.

Beranic, T., Rek, P., Heričko, M., (2020). *Adoption and Usability of Low-Code/No-Code Development Tools*. Central European Conference on Information and Intelligent Systems. <https://www.proquest.com/openview/a6e9a1210ef714ead2f9695c6a71fb6f/1?pq-origsite=gscholar&cbl=1986354>

Bäckström, M., Silversved, N., (2021) *Digitalizing the workplace: improving internal processes using digital services*  
<https://www.diva-portal.org/smash/get/diva2:1572255/FULLTEXT01.pdf>

Gaikwad, S. S., & Adkar, P. R. A. T. I. B. H. A. (2019). A review paper on bootstrap framework. *IRE Journals*, 2(10), 349-351.

Guntupalli, R.C.C. (2008). *User Interface Design - Methods and Qualities of a Good User Interface Design*. University West, Department of Technology, Mathematics and Computer Science.

Harrington, J. L. (2016). *Relational database design and implementation*. Morgan Kaufmann.

Kolonko, K. (2018). Performance comparison of the most popular relational and non-relational database management systems (Dissertation). Blekinge Institute of Technology, Faculty of Computing, Department of Software Engineering

Lefort, B., Costa, V., (2019). *Benefits of Low Code Development Environments on Large Scale Control Systems*. JACoW Publishing

Levene, M., & Loizou, G. (2012). *A guided tour of relational databases and beyond*. Springer Science & Business Media.

Luo, Y., Liang, P., Wang, C., Shahin, M., Zhan, J. (2021). *Characteristics and Challenges of Low-Code Development*, School of Computer Science, Wuhan University

Melton, J., & Simon, A. R. (1993). *Understanding the new SQL: a complete guide*. Morgan Kaufmann.

Meg, F. (2019). *Journal of Computing Sciences in Colleges*. Evansville, IN, United States

Okhrimenko, I., Sovik, I., Pyankova, S., Lukyanova, A., (2019) *Digital transformation of the socio-economic system: prospects for digitalization in society*.

<https://www.revistaespacios.com/a19v40n38/19403826.html#dos>

Rawat, P., & Mahajan, A. N. (2020). ReactJS: A modern web development framework. *International Journal of Innovative Science and Research Technology*, 5(11), 698-702.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2019). *Low-Code as Enabler of Digital Transformation in Manufacturing Industry*. Applied Sciences.

solid IT gmbh (2023). DB-Engines Ranking. Hämtad den 02. Mars 2023 från <https://db-engines.com/de/ranking>.

Sufi, F.,(2023). Algorithms in Low-Code-No-Code for Research Applications: A Practical Review. MDPI. <https://www.mdpi.com/1999-4893/16/2/108>

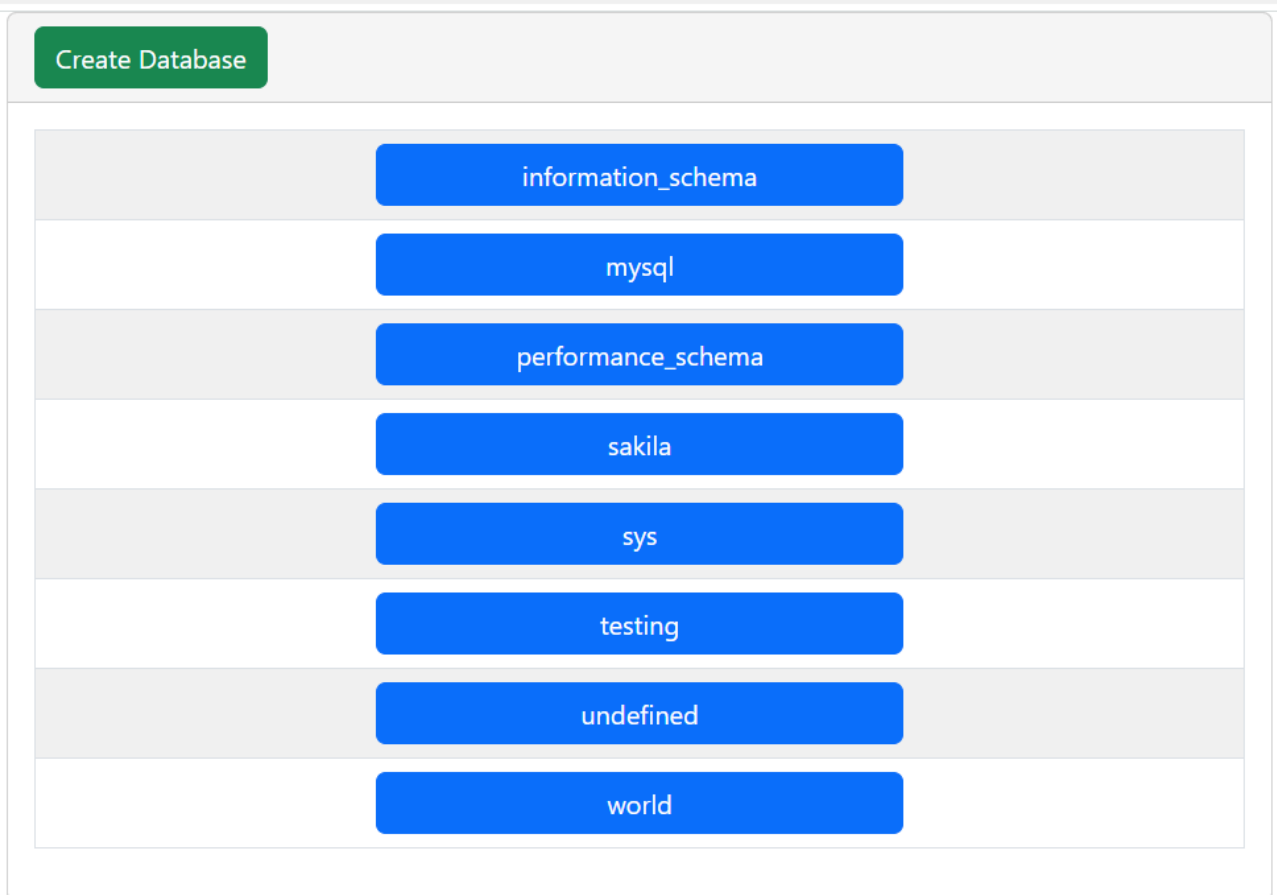
Tozzi, C. (2021) A practical take on low-code vs. traditional development 28/02/2023(<https://www.techtarget.com/searchsoftwarequality/tip/A-practical-take-on-low-code-vs-traditional-development#> )

Törnqvist, M. (2021). Kan applikationer utvecklas direkt ute i verksamheten?: En studie kring applikationsutveckling med low-code/no-code. Karlstad University, Faculty of Arts and Social Sciences (starting 2013), Karlstad Business School (from 2013).

## 11. Bilagor

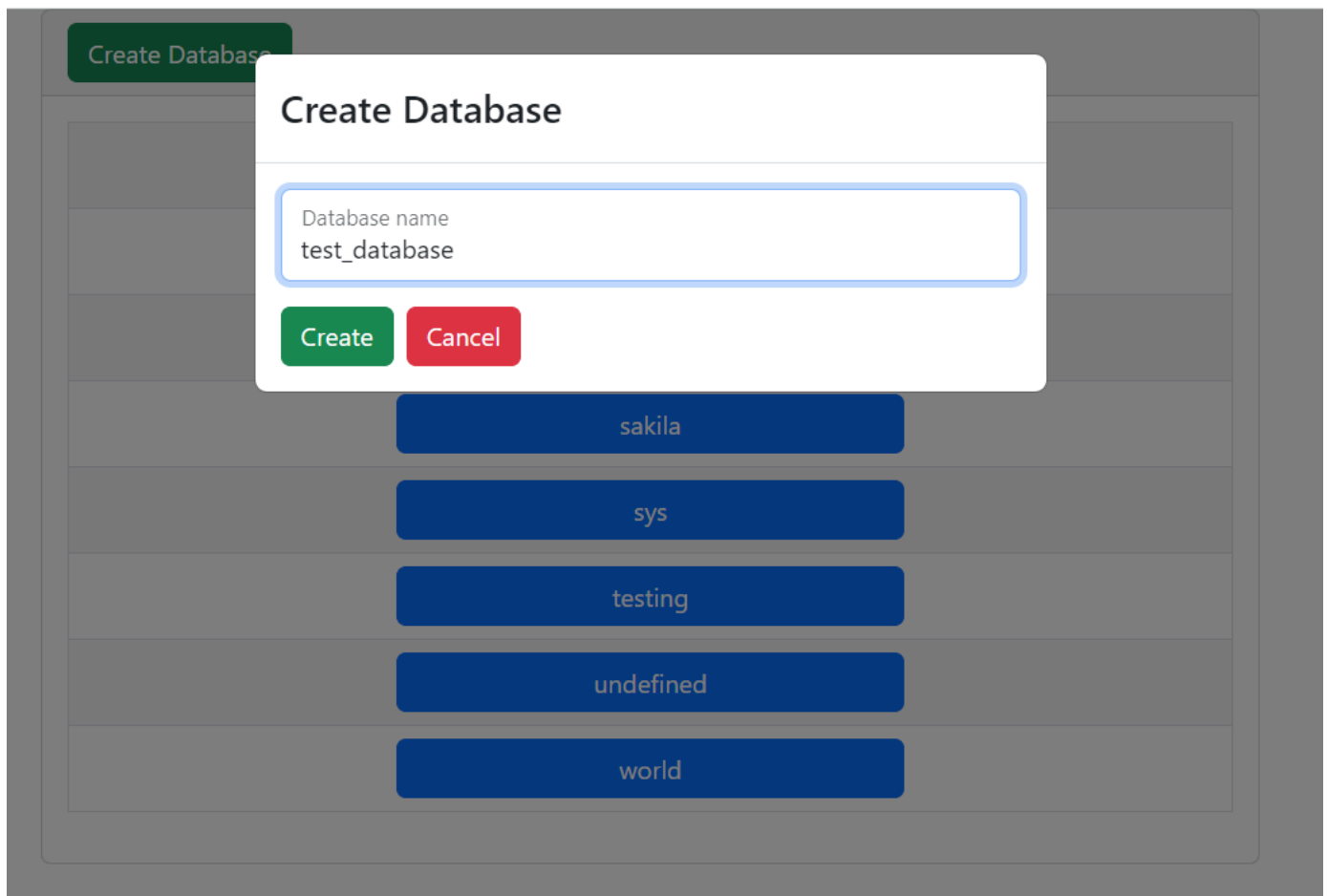
---

### Bilaga A



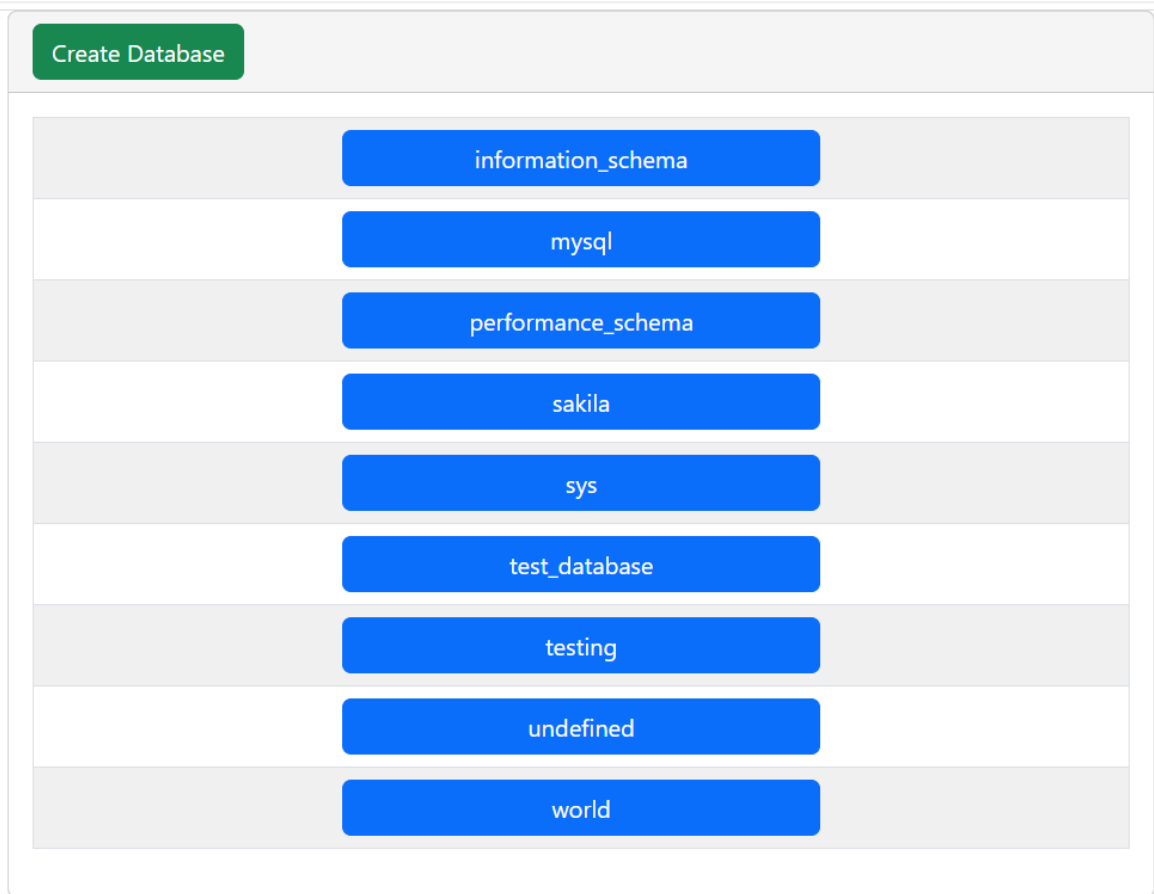
(Bild 2: Startsidan i vår applikation)

## Bilaga B



(Bild 3: Skapa ny databas med valfritt namn)

## Bilaga C



(Bild 4: Startsidan med ny skapad databas)



## Bilaga D

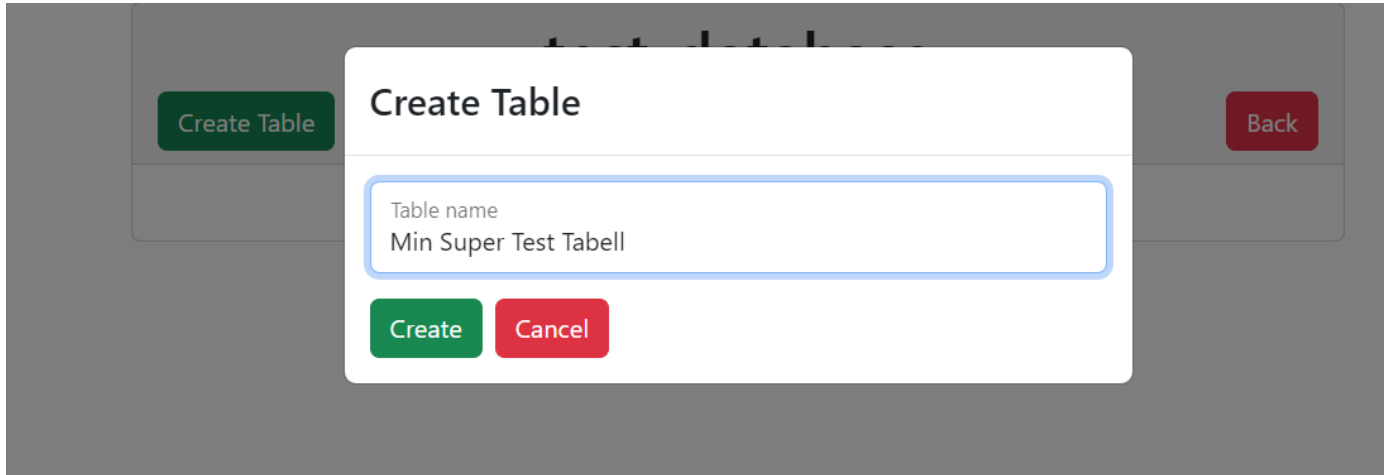
Create Table

test\_database

Back

(Bild 5: Databas sidan utan tabeller )

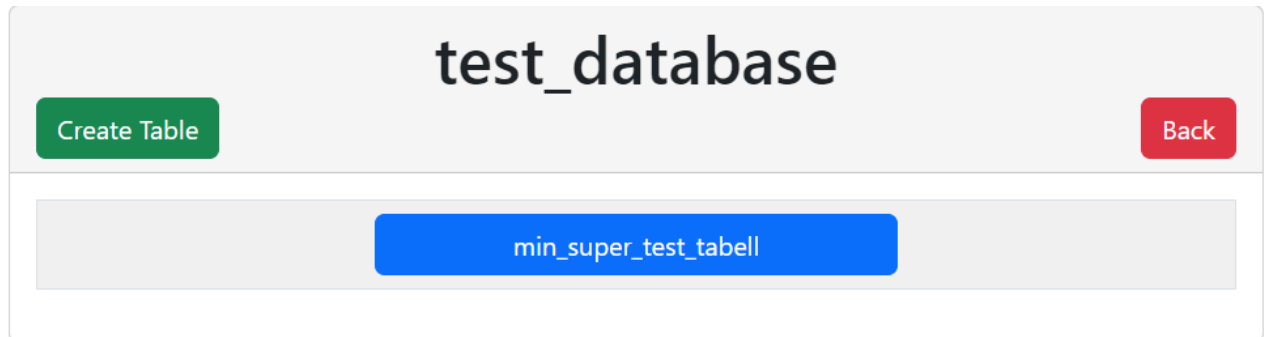
## Bilaga E



The image shows a 'Create Table' dialog box in the Newton database application. The dialog box is white with a light blue border and is centered on a dark gray background. It has a title bar that says 'Create Table'. Below the title bar, there is a text input field with the placeholder text 'Table name' and the entered text 'Min Super Test Tabell'. At the bottom of the dialog box, there are two buttons: a green 'Create' button and a red 'Cancel' button. In the background, a dark gray panel is visible with a green 'Create Table' button on the left and a red 'Back' button on the right.

(Bild 6: Skapa ny tabell på databasen)

## Bilaga F



(Bild 7: Sidan för databas med en nyskapad tabell )

## Bilaga G

min\_super\_test\_tabell

Create columnBack to mainBack

id

(Bild 8: Sidan på tabellen med initial kolumn id)

## Bilaga H

The screenshot shows a 'Create column' dialog box in the center. The dialog has a title bar 'Create columnn'. It contains two input fields: 'Name' with the value 'min\_test\_kolumn' and 'Datatype' with a dropdown menu showing 'Varchar (255)'. Below these fields is a checkbox labeled 'PrimaryKey' which is checked. At the bottom of the dialog are two buttons: 'Create' (green) and 'Cancel' (red). In the background, a table editor is visible with a column named 'id'. On the left of the background interface is a 'Create column' button, and on the right are 'Back to main' and 'Back' buttons.

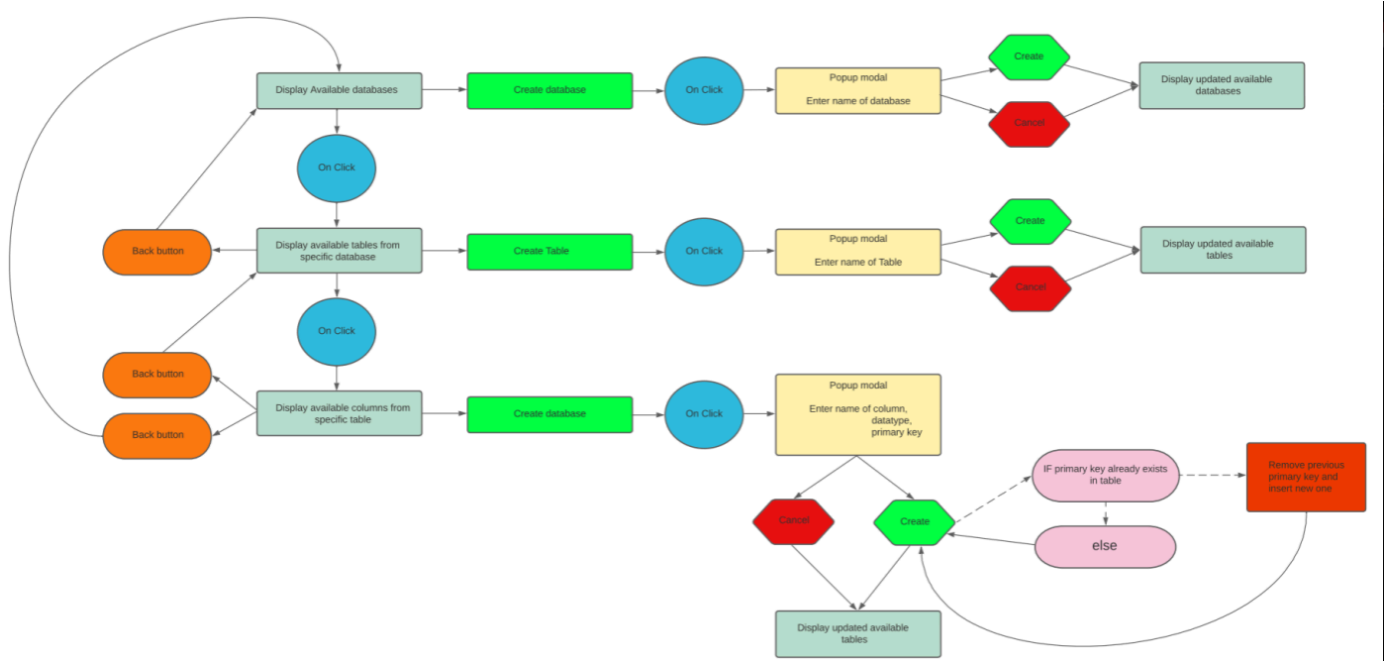
(Bild 9: Skapa en ny kolumn på tabellen)

## Bilaga I

min_super_test_tabell	
Create column	Back to main Back
id	min_test_kolumn

(Bild 10: Sidan på tabellen med nyskapad kolumn )

## Bilaga J



(Bild 11: Flowchart för vår applikation)