vera (model danych, wzorce stron, interfejs linkowo-przyciskowy) (UWAGA: termin oddawania do 19:00)

ZADANIE

Zadanie 2: aplikacja po stronie serwera (model danych, wzorce stron, interfejs linkowo-przyciskowy) (UWAGA: termin oddawania do 19:00)

Otwarto: niedziela, 11 kwietnia 2021, 00:00
Wymagane do: piątek, 5 maja 2023, 19:00

W tej części rozwoju aplikacji należy stworzyć model danych, odpowiednie strony z wzorcami do wypełnienia oraz interfejs linkowo-przyciskowy aplikacji.

Model danych

Należy stworzyć model danych z uwzględnieniem następujących encji:

- Katalog - encje tego rodzaju mają przechowywać informacje o plikach i innych katalogach. Oprócz opisu relacji z innymi encjami, mają:
  - nazwę,
  - opcjonalny opis,
  - datę utworzenia,
  - właściciela,
  - znacznik dostępności (fałszywy, gdy katalog został usunięty, początkowo wartość true)
  - datę zmiany znacznika dostępności,
  - datę ostatniej zmiany zawartości,
- Plik - encje tego rodzaju przechowują kod źródłowy programów. Z założenia kod źródłowy jest podzielony na znaczące sekcje. Oprócz opisu relacji z innymi encjami, encje tego rodzaju mają:
  - nazwę,
  - opcjonalny opis,
  - datę utworzenia,
  - właściciela,
  - znacznik dostępności (fałszywy, gdy katalog został usunięty, początkowo wartość true),
  - datę zmiany znacznika dostępności,
  - datę ostatniej zmiany zawartości.
- Sekcja pliku - encje tego rodzaju zawierają istotne znaczeniowo cząstki pliku lub komentarze; struktura sekcji jest złożona - sekcja może mieć podsekcje. Oprócz opisu relacji z innymi encjami, encje tego rodzaju mają:
  - opcjonalnie nazwę,
  - opcjonalny opis,
  - datę utworzenia,
  - początek sekcji,
  - koniec sekcji,
  - rodzaj sekcji,
  - status sekcji,
  - dane statusu,
  - treść.

Uwaga: do wyboru są dwa modele określania początku i końca sekcji. W pierwszym z nich sekcja zaczyna się zawsze na początku wiersza i kończy się na końcu. Wyznacznikiem początku i końca sekcji są tutaj numery wierszy. W drugim modelu początek i koniec sekcji może być w dowolnym miejscu. Wtedy miejsce te opisywane jest przez dwie wartości: number wiersza i numer znaku w wierszu. Można użyć dowolnego z tych modeli.

- Rodzaj sekcji - to encja, która określa typ zawartości sekcji; kategoria określa sposób, w jaki sekcja jest obsługiwana przez aplikację. Minimalny zestaw kategorii to: procedura, komentarz, dyrektywny kompilatora (#define, #pragma itp.), deklaracje zmiennych, wstawka asemblerowa.
- Status sekcji - to encja określająca stan sekcji; przykładowe statusy: kompiluje się z ostrzeżeniami, kompiluje się bez ostrzeżeń, nie kompiluję się.
- Dane statusu - dodatkowe dane związane ze statusem, np. błąd kompilacji, wskazanie w których linii dotyczy ostrzeżenie itp.
- Użytkownik – to encja, która określa użytkownika aplikacji. Oprócz opisu relacji z innymi encjami, encje tego rodzaju mają:
  - nazwę,
  - login,
  - hasło.

Powyższe encje oraz ich pola mogą mieć nazwy angielskojęzyczne.

Powyżej opisany model powinien mieć zrealizowaną obsługę za pomocą interfejsu użytkownika zgodnie z intencjami działania aplikacji.

Punktacja za encje

Za tablice realizujące poszczególne encje punkty przyznawane będą według schematu:

- Użytkownik - 1 p.,
- Encje związane z plikami i katalogami - 1 p.,
- Encje związane z sekcjami - 1 p.

Interfejs linkowo-przyciskowy i wzorce stron

Aplikacja powinna w następujący sposób udostępniać możliwość korzystania z możliwości kompilatora SDCC.

- Aplikacja powinna w sekcji *Wybór pliku* wyświetlać dostępną w bazie danych strukturę katalogów. Sekcja poiwnna być zaimplementowana jako odpowiedni fragment wzorca strony (lub wzorzec), który jest wypełniony danymi pochodzącymi z bazy danych.
- Aplikacja powinna umożliwiać dodanie pliku do bazy, podzielenie go na sekcje zgodnie z wymienionymi powyżej kategoriami (powinno się to dziać częściowo automatycznie, ale też powinna być możliwość określania sekcji ręcznie; w tym dodawania sekcji, oznaczania fragmentu pliku jako sekcji oraz łączenia istniejących sekcji w jedą wspólną sekcję). Operacje te powinny być dostępne w którymś z menu na pasku menu. Uwaga: pełna funkcjonalność tego typu możliwa jest do uzyskania dopiero po wprowadzeniu JavaScriptu, dlatego wystarczy, jeśli ten element pozostanie nie w pełni funkcjonalny i zrealizowana będzie jedynie część obsługi po stronie serwera.
- Aplikacja powinna umożliwiać dodanie katalogu do bazy. Operacja ta powinna być dostępna w którymś z menu na pasku menu.
- Aplikacja powinna umożliwiać skasowanie pozycji w katalogu (pliku, innego katalogu), przy czym plik czy katalog nie powinien być zupełnie usuwany z bazy danych, a jedynie należy go oznaczać jako niedostępny za pomocą znacznika dostępności. Operacja ta powinna być dostępna w którymś z menu na pasku menu oraz być połączona z odpowiednim mechanizmem wskazywania pliku lub katalogu.

- Aplikacja powinna w sekcji *Fragment kodu* wyświetlać zawartość pliku `filename.asm` uzyskanego w wyniku działania polecenia

```
# sdcc -S <filename>.c
```

  odpowiednio wzbogaconego przez inne opcje, jakie zostały określone w interfejsie użytkownika, przy czym `filename.c` odpowiada obecnie wybranemu plikowi. Wynik z pliku `filename.asm` powinien zostać podzielony na zakresy oddzielone kreskowanymi liniami. Przy czym najechanie myszką na sekcję powinno spowodować jej wyróżnienie. Osobny rodzaj wyróżnienia ma dotyczyć nagłówka sekcji, a osobny treści. Wyświetlanie tej sekcji powinno być zapewnione przez odpowiedni fragment wzorca lub wzorzec.
- W dolnej części ekranu powinny być udostępnione cztery taby. Oto opis ich zawartości
  - Pierwszy tabulator (zatytułowany STANDARD) powinien wskazywać, z jakim standardem powinna być zachowana zgodność kompilatora (co najmniej C89, C99, C11). Powinno być możliwe wybranie jednego z nich. Wtedy wszystkie wykonania kompilatora wykonywane przez aplikację powinny dotyczyć wybranego tutaj standardu.
  - Drugi tabulator (zatytułowany OPTYMALIZACJE) powinien zawierać listę dostępnych rodzajów optymalizacji (co najmniej 3). Powinna być możliwość określenia wybranego zestawu optymalizacji. Wtedy wszystkie wykonania kompilatora wykonywane przez aplikację powinny generować kod zgodnie z zestawem optymalizacji.
  - Trzeci tabulator (zatytułowany PROCESOR) powinien zawierać listę dostępnych w SDCC architektur procesorów (co najmniej MCS51, Z80 i STM8). Powinno być możliwe wybranie jednego z nich. Wtedy wszystkie wykonania kompilatora wykonywane przez aplikację powinny dotyczyć wybranego tutaj procesora.
  - Czwarty tabulator (zatytułowany ZALEŻNE) powinien zawierać listę opcji kompilatora, które są zależne od procesora. Dla każdego wybranego procesora powinna się tutaj znajdować możliwość wybrania trzech opcji właściwych dla niego (np. dla procesora MCS51 może się tutaj znajdować możliwość wyboru docelowego modelu programu - small, medium, large, huge model programs).
  - Menu powinno zawierać opcję pozwalającą na wykonanie pełnej kompilacji obecnie wskazywanego pliku. Przebieg powinien uaktualniać zawartość sekcji *Fragment kodu* oraz dawać możliwość zapisania wyniku kompilacji na lokalnym dysku.

Punktacja

- Powiązanie wyboru pliku w sekcji *Wybór pliku* z zawartością sekcji *Tekst programu* - 1 p.,
- Nawigacja po katalogach i plikach, dodawanie ich i usuwanie - 1 p.,
- Automatyczny i ręczny podział pliku na sekcje - 1 p.,
- Obsługa sekcji *Fragment kodu* - 1 p.,
- Prawidłowa obsługa wyboru standardu - 0,5 p.
- Prawidłowa obsługa wyboru optymalizacji - 0,5 p.
- Prawidłowa obsługa wyboru procesora i opcji zależnych od niego - 1 p.
- Prawidłowa obsługa kompilacji - 1 p.

Dodaj pracę

Status przesłanego zadania

Status przesłanego zadania	Nie przesłano jeszcze zadania

Stan oceniania	Nieocenione
Pozostały czas	Pozostało 31 dni 9 godzin
Ostatnio modyfikowane	-
Komentarz do przesłanego zadania	Komentarze (0)