Paulo Alexandre Duarte Ferreira

Licenciado em Engenharia Electrotécnica
pela
Faculdade de Engenharia da Universidade do Porto

GESTÃO INFORMÁTICA DE PATRIMÓNIO HISTÓRICO E ARTÍSTICO COM GEOREFERENCIAÇÃO

Faculdade de Engenharia da Universidade do Porto

1994

Tese realizada sob a orientação de

Prof. Dr. Fernando Nunes Ferreira

Professor Catedrático
do
Departamento de Engenharia Electrotécnica e de Computadores
da
Faculdade de Engenharia do Porto

Tese submetida para satisfação parcial dos requisitos

do

Curso de Mestrado em Engenharia Electrotécnica e de Computadores

(Perfil de Informática Industrial)

Resumo

Pretendemos com esta tese, descrever o estudo, especificação e implementação de um protótipo de base de dados para a inventariação dos azulejos da Cidade do Porto, e da sua integração com um sistema de informação geográfica. Este trabalho debruça-se principalmente sobre os problemas associados à informatização de museus e à associação de bases de dados com sistemas de informação geográfica. Os pontos discutidos incluem a preservação dos dados, a sua estruturação, a capacidade de crescimento do sistema, a sua integração na informática municipal, a escolha de uma plataforma para a implementação, de um ambiente de desenvolvimento, a interface com o utilizador, e a nossa visão sobre a evolução futura da informatização de museus.

Abstract

We aim to describe the study, specification and implementation of a database prototype for the inventory of tiles from the City of Porto, and the integration of the database with a geographical information system. We give the focus to museum informatics and to the association between the database and the geographical information system. Other areas also studied are data preservation problems, data structures, the system's scalability, the system's integration with the municipality's computer infrastructure, the choice of a proper implementation platform, the choice of the development system, the user-interface, and a panorama about the future evolution of the museum informatics.

Agradecimentos

À Dra. Maria João Vasconcelos, ao Arq. Guerra e à Dra. Maria Augusta Marques da divisão de Museus e Património Histórico e Artístico da Câmara Municipal do Porto, pela colaboração e pelo excelente trabalho desenvolvido na conservação e divulgação do património da cidade.

Ao Prof. Nunes Ferreira pela orientação e ajuda na elaboração deste projecto.

A todos os elementos do grupo de CG&CAD do INESC Porto, pela amizade e pelos conhecimentos partilhados.

Ao ISEP, pelo apoio à realização do mestrado.

A Ben Webster, Coleman Hawkins, Benny Carter, Dizzy Gillespie, Oscar Peterson, Stan Getz, Duke Ellington, Errol Garner, Ray Bryant, Sonny Stitt, John Coltrane, William Ackerman, Michael Hedges, George Winston, David Grisman e muitos outros.

Indíce

1. Apresentação do projecto

1.1 Introdução

Neste capítulo pretendemos descrever a origem deste projecto, as suas motivações, os seus objectivos, fornecer uma panorâmica sobre a informatização de museus, descrever os seus problemas, propor algumas soluções e apresentar o nosso trabalho.

1.2 Origem do projecto

Este projecto nasceu de contactos entre o grupo de CG&CAD do INESC Porto e a divisão de Museus e Património Histórico e Artístico da Câmara Municipal do Porto.

Neste momento, uma das prioridades da divisão, é a sua informatização para uma gestão mais racional e cuidada do património histórico e artístico da cidade. Como experiência piloto, de comum acordo entre as duas entidades, foi sugerida a realização de um protótipo de uma base de dados sobre os azulejos da cidade. Este protótipo servirá como ponto de partida para futuras realizações informáticas e para aumentar a colaboração entre os dois grupos.

A divisão de Museus e Património Histórico e Artístico da Câmara Municipal do Porto, tem um arquivo de azulejos e materiais cerâmicos, originários de prédios antigos que foram demolidos ou modificados. Este arquivo, destina-se a suportar a restauração de casas antigas, uma vez que certos azulejos e materiais cerâmicos, já não se fabricam. Um dos problemas da divisão é a inventariação eficiente do arquivo, que possibilitará uma gestão mais adequada, que começa a ser difícil, dado o seu sucesso e crescimento. Como a informação sobre os azulejos (e sobre o património da cidade) tem uma componente geográfica, isso levou a que o sistema concebido para a gestão do património incluisse uma componente de georeferenciação.

Ao longo desta tese vamos fornecer uma visão panorâmica sobre a informatização de museus, assinalar os factores únicos que influenciaram o nosso projecto, especificar um sistema adequado, descrever a sua funcionalidade, apresentar as

razões da escolha da plataforma e do ambiente de desenvolvimento, finalizando por expôr os detalhes mais significativos da implementação do sistema.

1.3 A Informatização de Museus e Arquivos Históricos

1.3.1 Problemas específicos

A informatização do inventário é problemática, se não tivermos em conta as características a que os programas de informatização de museus e arquivos históricos, devem obedecer, em termos de preservação dos dados, facilidade de conversão para outros formatos e uma boa funcionalidade para uma gestão eficiente do património [Bearman 94].

A informatização de museus encontra-se numa fase incipiente apesar de usar normalmente tecnologias avançadas. Devido à quantidade e ao tipo de dados envolvidos (imagens, etc), a informatização de museus tem de recorrer às últimas novidades tecnológicas. O atraso verificado na informatização de museus deve-se à evolução contínua das tecnologias que cria uma certa desconfiança nos responsáveis pela gestão dos museus.

A desconfiança dos conservadores em relação aos computadores, prende-se com os prazos de tempo previstos para a duração de um arquivo. Um computador ou um programa, ao fim de 10 anos de vida (por exemplo), encontram-se irremediavelmente ultrapassados e a necessitar de substituição. Um arquivo de fichas em cartão (papel não ácido), escritas com tinta adequada, com algum cuidado no seu manuseamento e num ambiente adequado, em termos de luz, humidade e temperatura, pode aguentar facilmente cerca de 400 anos.

Na estimativa de custo associada à informatização de um arquivo, além das verbas associadas à renovação periódica do hardware e software, devemos também associar verbas à renovação periódica do meio físico de armazenamento dos dados. Essa renovação pode ser efectuada, por outros meios do mesmo tipo ou por novos meios de armazenamento dos dados. Por exemplo: periodicamente devemos copiar os ficheiros em banda magnética para novas bandas, e talvez mudar para bandas de alta capacidade, se estivermos a usar bandas de capacidade reduzida.

Fazendo uma análise puramente económica, olhando apenas para os gastos, vemos então que os arquivos clássicos são muito mais baratos, a longo prazo, que os arquivos informatizados.

As grandes vantagens da informatização prendem-se normalmente com o processamento, a transmissão da informação e a sua maior acessibilidade. O processamento dos dados recolhidos tem uma grande importância, em qualquer actividade, para estudos e análises, sendo desnecessário tecer comentários sobre a sua utilidade. A transmissão da informação, por meios informáticos, tem vindo a conhecer uma grande evolução, que também se fez sentir na área museológica.

A distribuição de imagens e documentos, por via electrónica, vem no entanto, criar problemas legais, relacionados com a propriedade intelectual dos documentos e imagens, que se distribuem, e sobre a legitimidade da sua utilização para certos fins.

A preservação dos dados, como objectivo principal de um arquivo, é plenamente conseguida com a sua informatização. Depois da passagem da informação a uma forma digital, se usarmos métodos de armazenamento adequados e cuidarmos dos meios de armazenamento com uma renovação contínua, os dados poderão permanecer inalteráveis, ao longo de um prazo de tempo teóricamente infinito, uma vez que cada cópia digital é idêntica ao original, não se registando nenhuma degradação.

Na preservação de textos, as vantagens da informatização, em termos de preservação da informação, não são grandes em relação ao armazenamento de textos em papel, mas no armazenamento de imagens, as vantagens da informatização são notáveis.

O arquivo de imagens é um assunto problemático e difícil, sobretudo se quisermos preservar imagens coloridas [Brierly 93]. Normalmente, uma foto a preto e branco tem uma duração estimada entre 100 e 300 anos, dependendo do tratamento a que foi sujeita depois da revelação e das condições de armazenamento. Para fotografias "normais" a cores, o seu tempo de duração situa-se entre os 5 e os 20 anos. Os slides possuem uma longevidade maior, atingindo os 40 a 50 anos. Existem processos especiais capazes de fornecer fotografias com uma longevidade maior, mas o seu

custo é proibitivo. Um dos métodos utilizados é a separação de uma imagem a cores nas três componentes primárias, que são reproduzidas em três imagens a preto e branco. Outro método consiste na utilização de pigmentos extremente estáveis, utilizados normalmente na produção de tintas para a pintura automóvel, produzindo imagens a cores de uma excelente qualidade, mas a um preço de cerca de $500 US por fotografia. A exposição à luz, que encurta significativamente a vida de uma fotografia, é obrigatória se a quisermos observar ou expor, colocando normalmente os conservadores num dilema, uma vez que têm de escolher entre a divulgação das imagens e a sua preservação.

Por estas razões a digitalização de imagens, se for efectuada com um mínimo de qualidade, pode tornar-se num método económico e prático de conservação de imagens.

1.3.2 Tendências gerais

Os sistemas informáticos usados em museus, pertencem normalmente a duas categorias: as base de dados e os programas de "hipermédia" usados para produzir apresentações ou "quiosques" interactivos.

As bases de dados usadas são produtos comerciais, que um conservador ou especialista adapta, não existindo práticamente aplicações específicas, dada a dimensão reduzida do mercado e a enorme variedade de museus existente.

Os programas "hipermédia" ou apresentações multimédia, aparecem na maior parte dos casos, como experiências levadas a cabo por um grande museu, com o patrocínio de um mecenas, levando em linha de conta o impacto mediático de tais realizações.

O aspecto mais importante e prioritário na informatização de um museu ou arquivo prende-se com a reutilização da informação produzida. Se apenas produzirmos um programa de apresentação, sem estruturar a informação e recolhendo somente a informação necessária para o programa, então muita da informação terá o programa como destino e limite final. Por outro lado, se decidirmos estruturar a informação de uma forma adequada, e produzir uma base de dados apropriada, os resultados finais serão reutilizáveis, e adequados a uma posterior migração para qualquer tipo de documento. Isto, apesar de os resultados serem obtidos ao fim de uma escala de

tempo maior, e do impacto mediático ser menor. Uma base de dados poderá, então, funcionar como suporte para a produção de documentos multimédia ou hipermédia, diminuindo o tempo e o investimento necessários à produção destes.

Além da apresentação local dos dados, com o aparecimento das redes de computadores, e da sua interligação à escala global, surgiram aplicações para consulta de bases de dados remotas, que depois evoluíram até suportar imagens. O aproveitamento destas possibilidades por alguns museus, resultou na possibilidade de vermos, remotamente, por exemplo, imagens do Museu do Vaticano [Museum-L 94].

Este novo método de expôr material, terá a nosso ver um grande incremento, com o aumento previsto das redes de comunicações de dados de alto débito, com ligações intercontinentais, com uma utilização cada vez mais fácil e barata. Assim, no futuro poderemos "visitar" electronicamente os museus que quisermos, a qualquer hora, e examinando apenas as peças que nos interessem. Embora a curto prazo esta ideia seja utópica, pode-se tornar realista se a reduzirmos à interligação de museus, existindo em cada um além do seu património, equipamento informático e de telecomunicações ao dispor do público, que poderá ter acesso a outros museus, que de outra forma, dificilmente conheceria.

A distribuição clássica através de meios removíveis de armazenamento de dados, também sofreu uma revolução, através de novos meios de distribuição de informação, como os CD-ROM, que permitem obter capacidades elevadas de memória, a um custo reduzido.

1.3.3 Escolha da informação a armazenar

As primeiras bases de dados criadas para museus e a maioria das existentes hoje em dia, constituem aquilo que se pode chamar software para a documentação das colecções. Isto é, na base de dados estão os dados relativos à classificação e catalogação dos objectos, que foram obtidos apenas a partir do objecto, considerando-o como uma entidade isolada e que muitas vezes podem ser fornecidos por um simples exame (exemplo: cor, tamanho, etc), ou então dados extremamente subjectivos (exemplo: segundo um especialista um objecto pertence a uma categoria,

segundo outro pertence a outra, e segundo um terceiro nenhuma das duas categorias deve ser considerada, mas deve-se usar uma classificação alternativa).

Se analisarmos as razões que nos levam a preservar um objecto, veremos que o valor museológico de um objecto, tem a ver não com o objecto em si, mas com a sua origem e com todas as "entidades" ligadas ao objecto.

Numa perspectiva alargada de herança cultural, a origem de um objecto não é só a data e local de fabrico, mas inclui todas as relações que o objecto possuiu com:

- pessoas
- organizações
- culturas
- datas
- locais (georeferenciação)
- sítios arqueológicos
- prédios
- acontecimentos

Se quisermos aproveitar ao máximo as potencialidades da base de dados, esta também será usada para a gestão das colecções do museu. Esta assumiu um valor de primeiro plano, dadas as limitações de espaço existentes na maioria dos museus, que conduzem à realização de exposições temporárias, incorporando apenas uma parte do património total de um museu. Outro factor que aumenta a necessidade de um gestão cuidada é a vulgarização do empréstimo de peças entre museus.

Certos autores, apresentam até a gestão de colecções, e o planeamento e escalonamento de acontecimentos, como sendo a prioridade máxima de um conservador [Bearman 94]. Como na origem de um objecto, o importante a reter da gestão de um objecto são as suas relações, existindo uma curiosa simetria entre a origem de um objecto e a sua gestão (o seu futuro).

A gestão de um objecto vai ser efectuada com base na informação disponível acerca de:

•pessoas (funcionários)

•organizações

•datas e intervalos de tempo dispendido

•locais de armazenamento (georeferenciação)

•prédios (espaço envolvido)

•acontecimentos

Esta nova aproximação à informatização de museus, é mais adequada ao modo de funcionamento actual da maioria dos museus e além de ser uma ferramenta de documentação dos objectos, é uma ferramenta poderosa de apoio à gestão do museu.

1.4 O caso específico do projecto

As características que moldaram o projecto são muito específicas, inviabilizando o uso de software que não seja "feito à medida". A seguir, apresentamos uma lista dos factores que influenciaram a decisão de desenvolver software especificamente para este projecto:

•A limitação geográfica dos azulejos a Portugal.

•Apesar de existir no país um Museu do Azulejo, este está centrado nos azulejos do século XVII, e a maioria dos azulejos do Porto, com interesse histórico pertence aos séculos XVIII e XIX, tendo características totalmente diferentes.

•A integração do sistema com o sistema informático da Câmara Municipal, é quase única a nível de museus, uma vez que normalmente os museus não têm nenhuma relação com as câmaras (exceptuando as financeiras).

•Mesmo que um museu dependa de uma câmara, não é obrigatória a troca intensiva de informação entre as duas entidades.

•O sistema inclui a integração de uma base de dados com um sistema de informação geográfica.

•O sistema deve oferecer certas facilidades para extrair informação de imagens, muito difíceis de obter em bases de dados tradicionais.

•O sistema deve ser expansível em termos de capacidade, e número de utilizadores, podendo passar de um sistema baseado num computador pessoal com um único utilizador, para um sistema multiposto com acesso distribuído.

•A portabilidade dos dados deve ser assegurada, prevendo desde já que o sistema fique um dia obsoleto, tornando necessária a migração dos dados para outra plataforma.

1.5 Conclusão

As características do sistema são únicas, exigindo um estudo cuidadoso, uma vez que deverá funcionar a muito longo prazo e possuir uma enorme capacidade de crescer, sem atingir situações de saturação.

As vantagens da informatização foram provadas, e tentámos prever a evolução futura do software para museus e arquivos históricos.

Nos capitulos seguintes vamos descrever os requisitos a que o nosso sistema deve obedecer, a configuração e utilização do sistema realizado, justificar as escolhas efectuadas na sua constituição, descrever o ambiente de desenvolvimento utilizado, e os detalhes mais significativos da implementação do protótipo.

2. Especificação e descrição do sistema

2.1 Introdução

Neste capítulo vamos apresentar os requisitos a que o nosso sistema obedece, descrever configurações típicas, e as possibilidades que o sistema oferece.

O sistema foi concebido na plataforma Windows, usando o Microsoft Access como sistema gestor de bases de dados e o Mapinfo for Windows como sistema de informação geográfica. Para o seu desenho, tomamos em consideração as necessidades e as expectativas dos utilizadores, as características particulares que a informação a incorporar na base de dados possui, de modo a fornecer uma funcionalidade máxima.

Vamos de seguida fornecer uma lista de requisitos a que a base de dados deve obedecer, passando depois a descrever configurações de hardware adequadas ao trabalho com o nosso sistema, fazendo finalmente, a descrição das funcionalidades por ele oferecidas.

2.2 Requisitos da base de dados

A base de dados a implementar deve contemplar e facilitar:

a) a incorporação de imagens na base de dados. (Note-se que não pretendemos construir um documento multimédia, nem sequer hipermédia ou hipertexto, uma vez que as imagens são suficientes para a documentação dos azulejos e a estrutura da informação não é apropriada a uma representação em hipermédia [Nielsen 90]. A estrutura dos dados não se adequa à sua representação em pedaços independentes, interligados numa rede. No entanto, prevemos que a base de dados ofereça facilidades que permitam uma fácil incorporação dos seus elementos, em documentos multimédia ou hipermédia, como se pode ver na alínea e).

b) a sua posterior adaptação para a inventariação de outros tipos de objectos (ex: ferros forjados, montras comerciais, etc...), ou até de todo o inventário existente (ou a efectuar) do património histórico e artístico da cidade do Porto.

c) a incorporação, conversão e/ou consulta de outras bases de dados existentes noutros serviços do município, de maneira a evitar a duplicação de trabalho, e discrepâncias entre a informação a que os vários serviços da Câmara têm acesso.

d) a possibilidade de criação de interfaces "amigáveis" e diferentes conforme o perfil de cada utilizador.

e) a extracção fácil de elementos para a produção de documentos de qualquer tipo, transformando a base de dados numa plataforma para a construção de documentos.

f) a possível migração da base de dados para outras plataformas e/ou outros sistemas gestores de bases de dados, de forma a assegurar a preservação dos dados a longo prazo (em termos informáticos).

g) a sua interligação com sistemas de informação geográfica, para que os utilizadores tenham acesso à componente geográfica da informação existente.

h) a manutenção do modo de funcionamento normal dos serviços de património, através da simulação dos impressos existentes, de modo a diminuir o tempo de transição para as novas ferramentas de trabalho.

2.3 *Configurações típicas*

Para o funcionamento da nossa base de dados, necessitamos de hardware que poderá ir de uma plataforma mínima até um sistema sofisticado e complexo, com vários postos de trabalho, equipamento diversificado para a aquisição de imagens e ligação em rede.

Por razões que apresentamos no capítulo "Fundamentação técnica do sistema", escolhemos a plataforma Windows, o Microsoft Access como ambiente de desenvolvimento, e o Mapinfo como sistema de informação geográfica. Para esta

plataforma necessitamos de um computador PC compatível, com um mínimo de características que tornem o trabalho realizável. Algumas configurações possíveis serão as seguintes:

1) Sistema mínimo de desenvolvimento

- Computador 486 DX a 33 Mhz, 8 Mb de Ram, 200 Mb de disco.
- Placa Gráfica capaz de suportar uma resolução de 640 por 480 pixeis a 24 bits, e 800 por 600 pixeis a 16 bits de cor por pixel.
- Monitor policromático de 14 polegadas.
- Scanner de "mão" a cores para a aquisição das imagens.
- MsDos, Microsoft Windows 3.1, Access 1.1 e Mapinfo for Windows.

Efectuámos nesta plataforma a maior parte do desenvolvimento do nosso programa, apenas tendo dificuldades, na aquisição de algumas imagens, que ultrapassavam a largura máxima do scanner.

2) Sistema aconselhável de desenvolvimento

- Computador 486 DX2 a 66 Mhz, 16 Mb de Ram, 1 Gb de disco.
- Placa Gráfica capaz de suportar uma resolução de 1024 por 768 pixeis a 24 bits de cor por pixel.
- Monitor policromático de 17 polegadas.
- Scanner "desktop" A4, a cores para a aquisição das imagens.
- CD-ROM com capacidade de leitura de Photo-Cd's multisessão.
- MsDos, Microsoft Windows for Workgroups 3.11, Access 1.1 e Mapinfo for Windows.

Este sistema permitirá uma maior velocidade de processamento, devido ao uso de um processador mais rápido, uma memória maior e ao uso do Windows for Workgroups 3.11. A aquisição de imagens é mais fácil com o scanner "desktop" e temos a possibilidade de usar imagens armazenadas em PhotoCd. O monitor de um

tamanho maior em conjunto com a placa gráfica fornecerá uma maior qualidade de apresentação da imagem.

3) Sistema ideal de desenvolvimento

- Computador 486 DX4 a 100 Mhz, 32 Mb de Ram, 2 Gb de disco.
- Placa Gráfica capaz de suportar uma resolução de 1280 por 1024 pixeis a 24 bits de cor por pixel.
- Monitor policromático de 20 polegadas.
- Scanner "desktop" A4, a cores para a aquisição das imagens.
- Scanner de slides e negativos no formato 35 mm.
- CD-ROM de velocidade dupla com capacidade de leitura de Photo-Cd's multisessão.
- Impressora de sublimação a cores no formato A4
- MsDos, Microsoft Windows for Workgroups 3.11, Access 1.1 e Mapinfo for Windows.

Obteremos com este sistema o aumento na velocidade de processamento, uma aquisição de imagens mais fácil e económica com a ajuda do scanner de slides, e um acesso mais rápido a PhotoCd's. Os acréscimos do tamanho do monitor, e da resolução da placa gráfica são pequenos em termos de percentagem mas grandes em termos de funcionalidade. A impressora possibilitará a impressão de documentos com uma qualidade adequada.

4) Sistema mínimo para utilização

- Computador 486 DX2 a 66 Mhz, 16 Mb de Ram, 800 Mb de disco.
- Placa Gráfica capaz de suportar uma resolução de 640 por 480 pixeis a 24 bits, e 800 por 600 pixeis a 16 bits de cor por pixel.
- Monitor policromático de 14 polegadas.
- Scanner "desktop" A4 a cores para a aquisição das imagens.
- Unidade de tape tipo QIC para backup.

•MsDos, Microsoft Windows 3.1, Access 1.1 e Mapinfo for Windows.

Esta plataforma oferecerá uma funcionalidade razoável, a um preço aceitável constituindo uma excelente plataforma para a avaliação do protótipo.

5) Sistema aconselhável para utilização

•Computador 486 DX4 a 100 Mhz, 32 Mb de Ram, 2 Gb de disco.

•Placa Gráfica capaz de suportar uma resolução de 1024 por 768 pixeis a 24 bits de cor por pixel.

•Monitor policromático de 17 polegadas.

•Scanner "desktop" A4, a cores para a aquisição das imagens.

•Scanner de slides e negativos no formato 35 mm.

•CD-ROM de velocidade dupla com capacidade de leitura de Photo-Cd's multisessão.

•Unidade de tape tipo QIC para backup.

•MsDos, Microsoft Windows for Workgroups 3.11, Access 1.1 e Mapinfo for Windows.

Este sistema terá uma funcionalidade acrescida com o aumento da velocidade de processamento, possiblitando a digitalização económica de imagens com o scanner de slides e negativos. O aumento do tamanho do monitor, da resolução da placa gráfica e da capacidade do disco permitirá uma utilização cómoda do sistema.

6) Sistema ideal para utilização

•Servidor 486 DX2 a 66 Mhz, 32 Mb de Ram, 5 Gb de disco

•Postos de trabalho 486 DX2 a 66 Mhz, 32 Mb de Ram, com um disco reduzido.

•Placas Gráficas com suporte para uma resolução de 1024 por 768 pixeis a 24 bits de cor por pixel.

•Monitores policromáticos de 17 polegadas.

•Scanner "desktop" A3, a cores para a aquisição das imagens.

•Scanner de "slides" e negativos no formato 35 mm, com auto-alimentação.

•Scanner de transparências com suporte para formatos até 8 por 10 polegadas.

•CD-ROM de velocidade quádrupla com capacidade de leitura de Photo-Cd's multisessão.

•Impressora de qualidade fotográfica no formato A4.

•Unidade de tape de 8mm para backup.

•MsDos, Microsoft Windows for Workgroups 3.11, Access 1.1 e Mapinfo for Windows.

•Placas de interface de rede.

•Software de rede apropriado.

•Hardware para integração deste sistema na rede informática municipal.

Este sistema permitirá uma funcionalidade maior, sendo constituido por vários computadores, distribuindo-se os periféricos de aquisição pelos vários computadores, e diminuindo as especificações em termos de disco e de monitor, de alguns deles. A impressora possibilitará a elaboração de documentação de alta qualidade.

2.4 Descrição do sistema

2.4.1 Descrição da sua funcionalidade

Vamos fornecer de seguida uma descrição da funcionalidade oferecida pelo sistema. Na base de dados elaborada, existe uma forte separação entre a entrada de dados (carga da base de dados) e a sua consulta, devido aos detalhes existentes na informação, que devem ser preservados e verificados quando se faz o preenchimento da base de dados.

De um modo sucinto, vamos descrever a introdução de novos dados, através dos passos que o programa dá:

•Os dados correspondentes ao prédio são introduzidos, e se o prédio ainda não existir no inventário, é criada a entrada correspondente. Para verificar se o prédio existe ou não, apenas é necessário o preenchimento do seu endereço, sendo os restantes dados introduzidos apenas se os dados referentes ao prédio não estiverem incluídos na base de dados. A freguesia não precisa de ser introduzida, sendo o seu nome escolhido entre uma lista das 15 freguesias da cidade do Porto

•Os dados correspondentes ao painel e aos azulejos que o constituem, incluíndo as imagens, são introduzidos. Para uma maior comodidade, e para evitar erros na introdução de valores e normalizar classificações, os valores de certos campos encontram-se limitados a uma lista de valores admissíveis, possuíndo o utilizador apenas a hipótese de escolher um valor entre os que constam da lista. Os campos que se encontram limitados dessa maneira são o estado de conservação (bom, razoável ou mau), o tipo de objecto cerâmico (azulejo avulso, painel, telha, mosaico, painel cerâmico, ou painel de pastilha) e o subtipo (liso, padrão, relevo, figura avulsa e figurativo). O preenchimento do campo subtipo é efectuado apenas se o tipo receber o valor "azulejo avulso".

• Como ajuda ao prenchimento das cores que identificam um azulejo, o programa tem uma opção, onde a partir da cor de um ponto da imagem assinalado pelo utilizador, fornece uma lista de cores previamente definidas, que estão mais próximas da cor assinalada. Depois, o utilizador poderá escolher uma cor entre as cores sugeridas pelo computador ou definir uma nova cor, se achar que a assinalada não se encontra entre nenhuma das sugeridas pelo programa.

• Com os dados totamente introduzidos, o computador fornece uma lista dos azulejos que julga mais próximos dos que constituem o painel, deixando ao utilizador a decisão de criar azulejos, caso os azulejos que constituem um painel sejam de um modelo inexistente na base de dados.

A consulta da base de dados pode ser feita através da selecção (procura) de valores nos campos de certas entidades, ou usando a facilidade de construção gráfica de perguntas oferecida pelo Access (QBE-Query by Example), que é bastante poderosa.

Uma das facilidades que a base de dados oferece é o processamento automático de épocas ou anos. Isto é, nos campos correspondentes a datas o utilizador pode escrever por exemplo 1845 ou Séc XX ou 1ª metade do séc. XIX e o programa aceitará esses valores.

A interligação ao sistema de informação geográfica, permite que visualizemos num mapa, a localização de um prédio seleccionado na base de dados. O sentido da transferência de informação pode-se inverter, possibilitando a escolha de vários prédios (ou apenas um) na base de dados, através da sua selecção no sistema de informação geográfica.

2.4.2 Descrição de alguns aspectos da interface com o utilizador

A base de dados apresenta uma interface extremamente simples, baseada em janelas múltiplas de um tamanho reduzido. Assim minimizamos o espaço de ecrã ocupado, deixando espaço livre para as outras aplicações, que serão utilizadas ao mesmo tempo que a nossa base de dados. O menu principal encontra-se colocado numa janela de orientação vertical (Figura 1).

Figura 1 O menu principal

As opções prédios, azulejos, paineis e imagens (Figura 2, Figura 3 e Figura 4) permitem abrir janelas correspondentes a cada uma das entidades. Essas janelas possuem um comportamento bastante sofisticado,uma vez que se tivermos a janela dos prédios aberta e abrirmos a dos paineis, apenas conseguiremos observar os paineis correspondentes ao prédio que se encontra na janela respectiva. Se mudarmos para outro prédio usando os botões da respectiva janela, os paineis que poderemos observar corresponderão ao novo prédio.

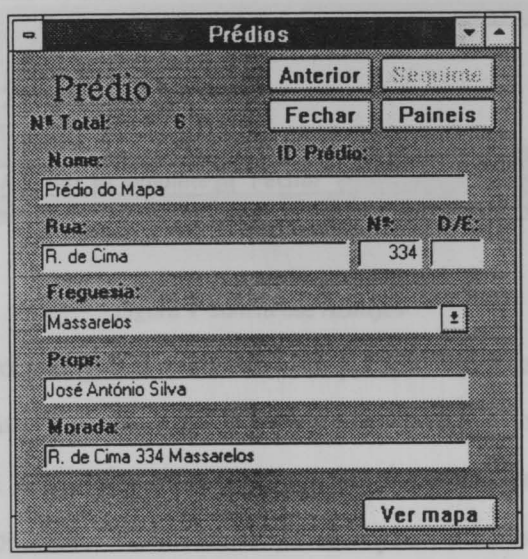

Figura 2 Janela dos prédios

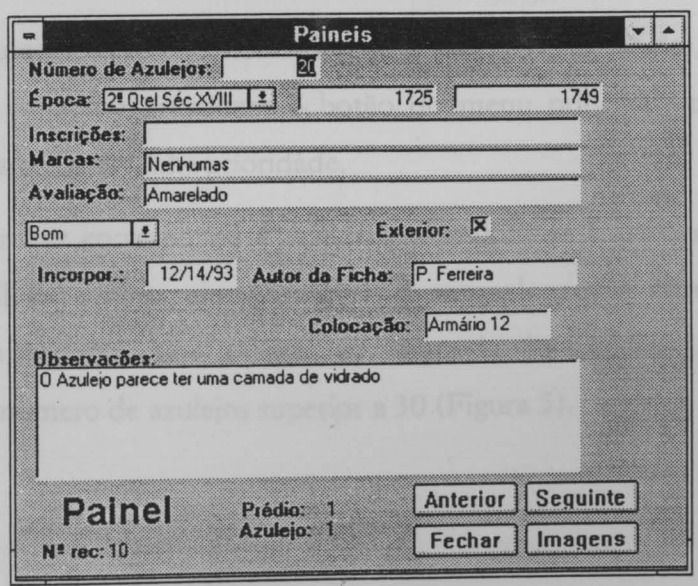

Figura 3 Janela dos paineis

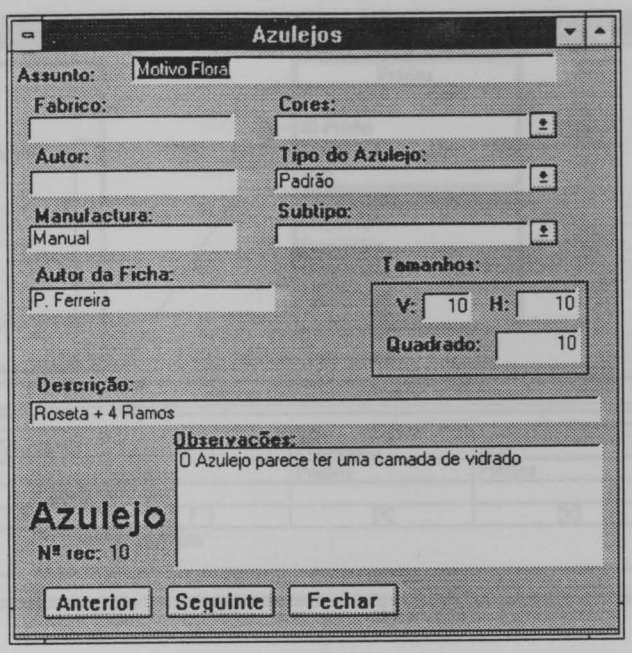

Figura 4 Janela dos Azulejos

As relações de prioridade que determinam se podemos visualizar ou não todas os objectos correspondentes a uma janela, ou se esta está limitada à visualização dos objectos que possuem uma relação com outra janela, prendem-se com a escala de antiguidade na sua abertura. Assim, uma janela que tenha sido aberta depois de outra, apenas visualizará objectos relacionados com o objecto visualizado na primeira janela.

Para desfazer uma relação de prioridade podemos fechar as janelas, tornando-as a abrir na ordem certa, ou pressionar o botão do menu principal corespondente à janela que desejamos que tenha prioridade.

A opção Interrogar corresponde à construção gráfica de interrogações, bastante poderosa e fácil de utilizar, demonstrando o exemplo como obter o nome do proprietário e o endereço dos prédios, da freguesia de Cedofeita, que possuem paineis com um número de azulejos superior a 30 (Figura 5).

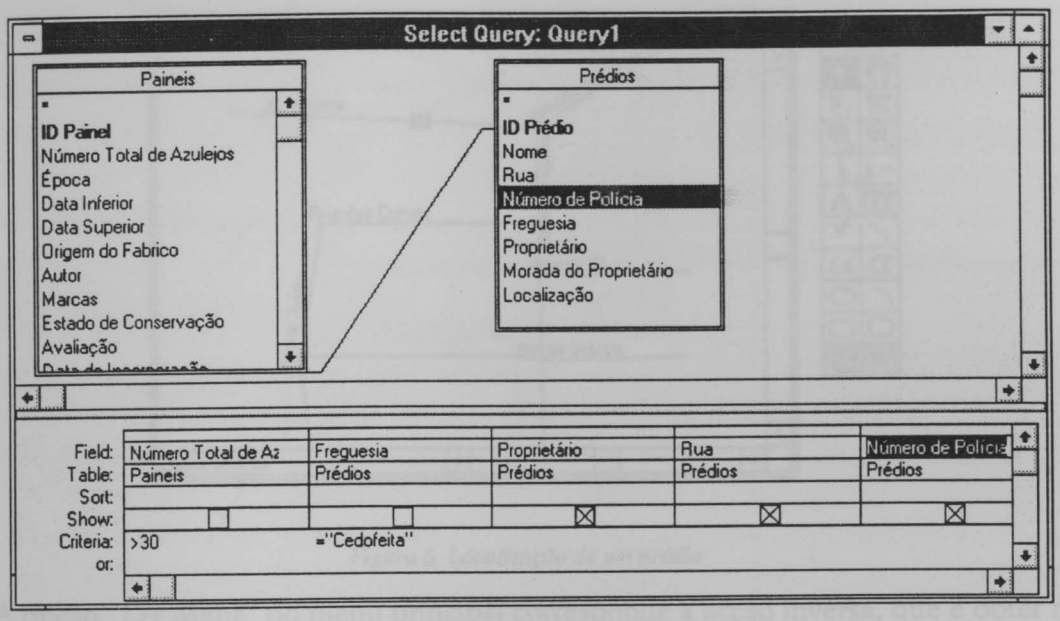

Figura 5 A construção de interrogações

Para utilizar a construção de interrogações, escolhemos primeiro as tabelas que necessitamos para a nossa interrogação. As tabelas escolhidas ficam colocadas na janela superior, com as relações que existem entre elas assinaladas pelo Access. Para construir uma interrogação, arrastamos para a janela inferior os campos que iremos utilizar, e assinalamos se os queremos visualizar na tabela de resultados ("show") e qual o critério para a sua selecção.

Na janela correspondente aos prédios (Figura 2) encontra-se um botão "Ver mapa". Esse botão tem por função assinalar no mapa do Sistema de Informação Geográfica a localização do prédio, como se pode ver na figura que ilustra essa função do programa (Figura 6).

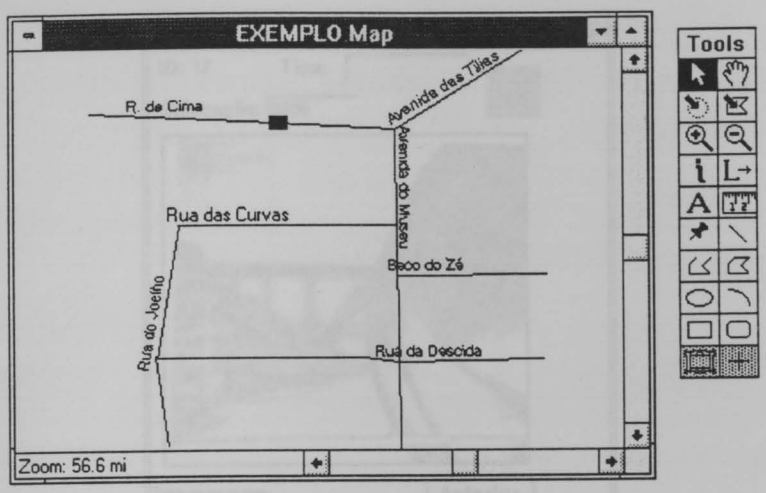

Figura 6 Localização de um prédio

A opção "Ler Mapa" do menu principal corresponde à acção inversa, que é obter na base de dados uma lista dos prédios correspondente a uma determinada selecção geográfica, efectuada previamente no sistema de informação geográfica (Figura 7).

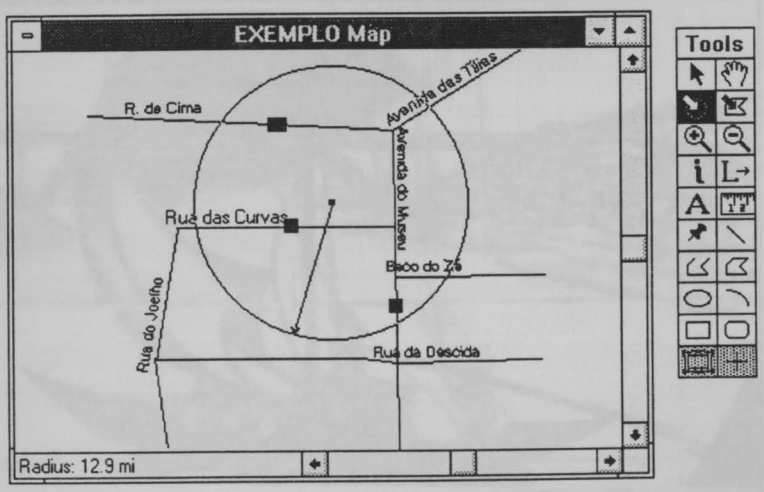

Figura 7 Selecção geográfica de vários prédios

Como resultado da operação "ler mapa", teremos na janela "prédios" da base de dados, o conjunto de prédios seleccionados no mapa. A partir dos prédios poderemos depois obter toda a informação, que desejamos.

A janela correspodente às imagens possui dois estados: normal (Figura 8), para poupar espaço de ecrã, e expandida (Figura 9), para possibilitar uma visão mais alargada das imagens. A comutação entre os dois modos, é feita usando um icon que simula um livro que podemos fechar ou abrir.

Figura 8 Imagem normal

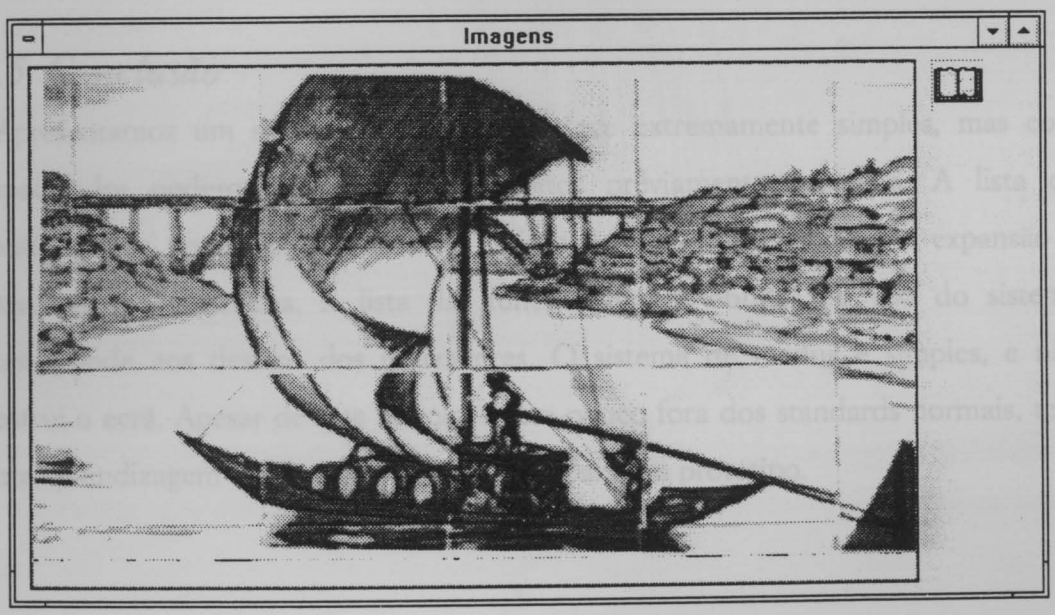

Figura 9 Imagem expandida

A opção seleccionar faz aparecer um submenu (Figura 10) a partir do qual podemos passar às janelas correspondentes aos prédios, azulejos, paineis, ou imagens, encontrando-se as janelas com os valores todos em branco, oferecendo uma facilidade de selecção dos registos a partir dos valores que preenchermos (uma espécie de procura).

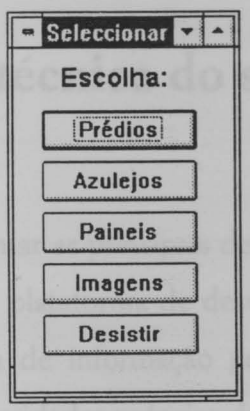

Figura 10 Menu Seleccionar

A opção impressos do menu principal acciona os procedimentos, de introdução de novos dados e impressão de formulários.

2.5 Conclusão

Apresentamos um sistema com uma interface extremamente simples, mas com capacidades poderosas, dados os requisitos préviamente expostos. A lista de configurações possíveis fornece uma ideia sobre as possibilidades de expansão e crescimento do sistema. A lista das funcionalidades obtidas no uso do sistema corresponde aos desejos dos utilizadores. O sistema de menus é simples, e não obstrui o ecrã. Apesar de este se colocar um pouco fora dos standards normais, tem uma aprendizagem rápida e fácil, sendo ideal para um protótipo.

3. Fundamentação técnica do sistema

3.1 Introdução

Neste capítulo vamos fundamentar as principais decisões que tomámos, na escolha do modelo da base de dados, da plataforma de desenvolvimento, do sistema gestor da base de dados e do sistema de informação geográfica. Estas decisões foram orientadas pelas restrições e necessidades próprias do sistema, que têm muito pouco em comum com os objectivos que normalmente orientam a escolha de um sistema informático.

3.2 Escolha do tipo da base de dados

Do ponto de vista académico, a implementação mais sedutora da base de dados seria uma estrutura orientada aos objectos [Parsaye 89]. No entanto, dadas as necessidades de portabilidade dos dados, interligação com software comercial já existente e facilidade de implementação, escolhemos o modelo relacional, tendo em conta a sua simplicidade e a penetração no mercado. Note-se que estamos a falar do modelo relacional no sentido "comercial" do termo e não no verdadeiro sentido do termo, porque iremos analisar o SGBD[1] a escolher, de forma a que os seus desvios do modelo relacional não sejam importantes para a nossa aplicação. Não consideramos sequer os modelos hierárquicos e em rede devido às suas caracteristicas inadequadas para a aplicação em vista [Ullman 89] e à escassez de software desse tipo em comparação com os SGBD´s relacionais. Além disso, todas as recomendações sobre a informatização de museus, apontam para o uso de modelos relacionais.

3.3 Escolha da plataforma de desenvolvimento

A plataforma de desenvolvimento deste protótipo foi escolhida por razões técnicas e económicas. Do ponto de vista económico, tentou-se minimizar tanto as despesas necessárias à execução deste protótipo, como as despesas futuras de integração e

[1] Sistema gestor de bases de dados

adaptação de um sistema totalmente funcional. A plataforma Windows foi escolhida tendo em linha de conta a existência de máquinas adequadas, o seu baixo preço, a facilidade de utilização, a grande variedade de software existente, a facilidade de comunicação entre dois programas Windows e, sobretudo, a facilidade e baixo preço da sua interligação com outro tipo de máquinas.

A escolha do software de desenvolvimento foi mais difícil devido à enorme variedade de métodos de implementação de bases de dados na plataforma Windows. Além disso as API´s[2] do Windows estão em mutação constante e os toolkits de programação estão a ser revistos, o que vem trazer uma grande incerteza relativamente ao estilo de programação "linguagem + chamadas à API". Isto faz com que este estilo de programação considerado normalmente mais "estável" em termos de futuro, seja instável, além de difícil dado o tamanho da API, se quisermos aproveitar ao máximo as potencialidades do Windows.

A programação em Windows complica-se ainda mais quando consideramos o OLE[3], que virá no futuro a sustituir a API normal das várias plataformas Windows [Britton 93].

Vamos esclarecer melhor os dois parágrafos anteriores. Neste momento (Fevereiro de 94) podemos ignorar o Windows NT, uma vez que se trata de uma plataforma de topo de gama, que irá ser utilizada fundamentalmente como servidor, devido às suas características de segurança e desempenho, que necessitam de um computador rápido, com uma grande memória e um disco rápido [DeVoney 93a] [Kennedy 93], [DeVoney 93b].

No entanto, nenhuma outra versão do Windows, se lhe pode comparar em termos de fiabilidade e segurança. Mesmo a futura versão 4.0 do Windows, não terá as facilidades de "multitasking" seguro, em que todas as aplicações estão protegidas umas das outras, nem será um sistema operativo de "32bits", restando-lhe ainda vestígios do MsDos e do BIOS. A maior vantagem do Windows 4.0 será uma nova

[2] Aplication Programming Interface - Interface oferecida para a programação por um sistema ou programa, que constitui o lado visível para o programador, na sua tarefa de construção de aplicações.

[3] Object Linking and Embedding - Método de comunicação entre aplicações Windows, usado inicialmente apenas para a construção de documentos compostos, resultantes da inclusão de documentos de uma aplicação em documentos de outra aplicação.

shell que substituirá o Program Manager, o File Manager e o Task Manager do Windows 3.1, integrando as três funcionalidades [Udell 94b]. Outra das vantagens do Windows NT é o facto de um programador não ter mais que se preocupar com os "segmentos" do Windows 3.1 e, acima de tudo o facto do Windows NT ser um sistema operativo portável, que corre em vários processadores (na família 386 Intel, Alpha, MIPS R4000 e Clipper RISC, estando anunciada uma versão para PowerPC). Para correr razoavelmente o Windows NT, necessitamos de uma máquina com um 486DX a 33Mhz, 16 Mb de memória e um disco de cerca de 300Mb, o que torna o Windows NT uma plataforma relativamente cara de momento.

As duas versões do Windows que restam (actualmente) são o Windows 3.1 e o Windows for Workgroups 3.11. O Windows for Workgroups é a versão recomendada pela Microsoft para uma utilização normal, enquanto que o Windows 3.1 é a versão aconselhada pela Microsoft para computadores 286 [Microsoft 94].

As diferenças entre o Windows For Workgroups (WfW) e o Windows estão centradas no acesso ao disco e aos periféricos de rede (inexistentes no Windows). O acesso ao disco no WfW já é feito através de "device-drivers" de 32 bits, que evitam a comutação entre modo real e modo protegido necessária no Windows normal, e que não usam o MsDos ou o BIOS, conseguindo-se assim um aumento de considerável nalgumas tarefas. A camada de software de rede no WfW funciona também a 32 bits, eliminando mais uma vez comutações de modo de funcionamento que são lentas. A característica mais inovadora do WfW é a possibilidade de utilizarmos o DDE[4], não só entre dois programas na mesma máquina, mas entre duas máquinas, o que vem abrir muitas possibilidades de comunicação entre software, que à partida estaria limitado a trabalhar "stand-alone" [Udell 94a], [Stephens 94].

Neste momento as razões pelas quais a Microsoft recomenda o WfW prendem-se com o seu desempenho (além do interesse monetário) e, no futuro, o Windows 4.0 incorporará todos os melhoramentos do WfW, em termos de acesso ao sistema de ficheiros [Udell 94b].

[4] Dynamic Data Exchange - Método de comunicação entre duas aplicações Windows, através de memória partilhada, que constitui o método mais antigo de comunicação inter-processos em Windows, uma vez que tem a sua origem no Windows 2.0.

No que diz respeito à programação em ambiente Windows, esta multiplicidade de versões vem trazer uma certa confusão. A Microsoft diz que existe apenas uma API para programação em Windows a 32 bits (substituindo a API antiga), chamada Win32S e que, seguindo certas regras se consegue fazer um executável que corra em Windows 3.*, Windows 4.0 e Windows NT. No entanto, a Microsoft também diz que o API do Windows 4.0 implementa uma grande parte das funcionalidades do Windows NT, e aumenta essas funcionalidades nalgumas àreas. Esse acréscimo de funcionalidade será oferecido daqui a algum tempo no Windows NT [Microsoft 94].

Aquilo que podemos afirmar neste momento é que a definição de um API do Windows necessita de alguma clarificação, se quisermos fazer um programa com um mínimo de futuro. Além da indefinição do API, a Microsoft diz que no futuro, a melhor maneira de programar em Windows é através do OLE 2.0 [Britton 93], [Heinen 94].

O OLE (Object Linking and Embedding) apareceu com o Windows 3.1, como uma maneira de inserir documentos de uma aplicação dentro de documentos de outra aplicação e, neste momento, constitui com a versão 2.0, o início da transição do Windows para um sistema orientado aos objectos [Microsoft 92]. Ainda é impossível construir uma aplicação, baseando-nos apenas no OLE 2.0 sem utilizar a API do Windows, mas uma grande parte da funcionalidade de uma aplicação pode ser implementada com o OLE 2.0 [Brocskschmidt 93].

O OLE tem um grande interesse como forma de comunicação entre aplicações, suplantando o DDE, apesar de um grande número de aplicações ainda não suportar o OLE 2.0 e o DDE ser uma tecnologia bastante robusta. Mas a funcionalidade do OLE é muito maior, conduzindo a uma automatização mais fácil das aplicações, e eliminando o assincronismo da comunicação por DDE, que pode provocar comportamentos imprevisíveis dos programas. No entanto, o OLE só funciona entre programas que correm na mesma máquina, enquanto que o DDE pode funcionar entre duas máquinas diferentes e, para programação normal, os objectos do OLE não se identificam muito bem com os objectos do C++ [Andrews 93], [Miller 94], [Kiyooka 94], [Kennedy 94].

As maiores desvantagens do OLE são o facto de adicionar cerca de 400 novas funções ao API do Windows, o facto de ignorar os "standards" propostos pelo Object Management Group[5] e as limitações de desempenho que um documento OLE faz aparecer, ao exigir que um computador tenha memória suficiente, para a soma dos programas que criaram esse documento [Berst 94].

No futuro, prevê-se que uma grande parte da programação em Windows consista na customização de aplicações, usando por exemplo o VBA (Visual Basic for Applications) como linguagem de programação e usando o OLE 2.0 para a comunicação entre as várias aplicações [Pleas 93].

À data, as linguagens de programação mais utilizadas para programar em ambiente Windows são da Microsoft: o Visual C++ 1.5 e o Visual Basic 3.0. Note-se que o adjectivo "visual" é puramente "publicitário", uma vez que nenhuma das duas linguagens é uma linguagem visual de programação.

Aquilo que as duas linguagens permitem é a construção gráfica da interface e a libertação do programador de uma das mais árduas tarefas: a detecção de eventos. Uma característica que ajudou o Visual Basic a captar uma grande fatia de mercado é o facto de se poder construir a interface com objectos originais da Microsoft ou com objectos fornecidos por outros fabricantes de software que podem possuir o comportamento adequado ao nosso programa. Actualmente, o Visual Basic tem um sucesso maior do que o Visual C++ devido à sua maturidade (existe há mais tempo), ao facto de uma aplicação típica necessitar de menos linhas de código e ao tamanho gigantesco do Visual C++ 1.5 (mais de 100Mb). É claro que uma aplicação escrita em Visual C++ vai ser mais rápida, mas na maioria das aplicações, o aumento da velocidade não é notado ou o programador prefere a diminuição do tempo de desenvolvimento.

[5] Object Management Group - Grupo de fabricantes de software que está a tentar produzir especificações para a criação e manipulação de documentos compostos, em várias plataformas. Inclui a IBM, Borland, Apple, Novell, e mais alguns grandes nomes do software.

3.4 Análise dos vários tipos de software para gestão de bases de dados

Podemos dividir as bases de dados relacionais para microcomputadores em três tipos: SGBD de "desktop", SGBD´s cliente-servidor e "front-ends" para bases de dados. A distinção que fazemos entre os três tipos é a seguinte:

- Um SGBD "desktop" é um software destinado a criar aplicações de bases de dados que utilizam ficheiros de uma única estrutura. O Paradox, o FoxPro, o dBase e o Clipper são exemplos de SGBD´s programáveis de "desktop". Estes produtos usam ficheiros individuais para as suas tabelas, índices e aplicações específicas. Alguns SGBD´s suportam dialectos de SQL[6] para a criação de interrogações, como complemento às suas linguagens de programação próprias. Mesmo que exista um servidor para partilha de ficheiros, todas as manipulações de ficheiros são feitas na máquina em que o programa corre. A maioria dos SGBD do tipo "desktop" não oferece qualquer tipo de segurança, sendo esta oferecida normalmente pelo software de rede. A maioria dos SGBD's do tipo "desktop" vem acompanhada de uma versão "run-time" sem limites de distribuição, para que os programadores possam distribuir cópias do seu software sem ter de pagar uma licença de distribuição por cada cópia.

- Um SGBD cliente-servidor possui uma estrutura do ficheiro de base de dados proprietária, para instalação num computador que funcionará como servidor. Os servidores de SQL da Microsoft e da Sybase, o Oracle e o Rdb são implementações típicas do modelo cliente-servidor. As tabelas, os índices e as aplicações de baixo nível encontram-se armazenadas num único ficheiro. Todos os produtos cliente-servidor usam o SQL para criar, modificar e interrogar tabelas da base de dados. As interrogações são processadas pelo servidor e o seu resultado é devolvido aos clientes pela rede. A segurança da base de dados é mantida pelo SGBD usando instruções SQL. As aplicações são criadas

[6] Structured Query Language - Linguagem standard de utilização e definição de bases de dados relacionais, usada principalmente para o processamento de interrogações.

normalmente com um software de um fabricante diferente, uma vez que as capacidades de manipulação e visualização de dados de um SGBD cliente-servidor, são bastante limitadas na maioria dos casos. O preço do software baseia-se normalmente no número de postos de trabalho, que podem aceder simultaneamente ao servidor do SGBD. O seu custo é elevado se adicionarmos o software e hardware, dos clientes, do servidor e da interligação entre eles. Mesmo que coloquemos o servidor numa plataforma PC ainda teremos problemas de preço e sérias limitações de performance na maioria dos servidores para PC [Butler 93]. Note-se que existem ferramentas poderosas e sofisticadas para a criação de aplicações clientes de SQL como o "Informix-Hyperscript for Windows" e "Oracle Cooperative Development Environment", mas o seu preço anda entre os $1500 e os $3000 US [Canter 93].

- Um "front-end" para base de dados é uma aplicação que fornece acesso às estruturas de ficheiros de diferentes SGBD´s. Podemos assim criar aplicações de manipulação de dados e visualização de dados, que utilizam tabelas de vários SGBDS´s cliente-servidor ou "desktop". Exemplos de "front-ends" são o PowerBuilder, Forest and Trees e o Q+E Database Editor. Todos os "front-end´s" suportam os comandos das linguagens do SQL para interrogação, manipulação de dados e processamento de transacções (DQL, DML e TPL); alguns conseguem ainda criar novas tabelas através da linguagem de definição de dados do SQL (DDL). Alguns "front-end´s" são destinados ao utilizador final e usam um linguagem para criação de "scripts" bastante simples, enquanto outros possuem uma linguagem de programação bastante completa, normalmente com uma sintaxe proprietária (não standard). Os "front-end's" cliente-servidor necessitam normalmente de uma licença por cada máquina onde o software está instalado e, normalmente, os mais económicos apenas permitem a consulta da base de dados [Watterson 94].

A distinção entre estes três tipos de SGBD´s tem vindo a ser destruida pelos últimos produtos lançados no mercado que, segundo os fabricantes, conseguem funcionar como SGBD´s "desktop", como cliente de SGBD´s remotos e como "front-end´s".

Para a nossa aplicação necessitamos de um produto que, numa fase inicial, funcione como um SGBD "desktop", para diminuir custos e facilitar a construção do protótipo, mas que possa no futuro funcionar como cliente SQL e integrar na sua base de dados tabelas remotas.

As bases de dados programáveis para o ambiente Windows mais difundidas são [Browning 93]:

- O Superbase, que foi uma das primeiras bases de dados a surgir no ambiente Windows, tem uma boa interface com o utilizador e uma curva de aprendizagem suave, uma vez que quase todos os comandos dos menus podem gerar o código correspondente às suas acções, o que transforma a produção de programas simples numa actividade fácil. O Superbase tem limitações relativamente à manutenção da integridade referencial nas suas bases de dados e um preço elevado, se incluirmos o "developers kit".
- O Paradox para Windows é uma plataforma atractiva de desenvolvimento, com uma linguagem de desenvolvimento fortemente orientada aos objectos, com um excelente ambiente de desenvolvimento e suporte para OLE e DDE. Note-se que o Paradox para Windows é muito diferente do Paradox para Ms-Dos, tem limitações de desempenho, um preço elevado, e, na altura da nossa decisão, apenas podia aceder a ficheiros dBASE e Paradox, através dos drivers IDAPI[7].
- O FoxPro para Windows é a base de dados mais rápida existente para Windows, oferece compatibilidade com o formato dBASE, muitas ferramentas de desenvolvimento, mas só pode processar tabelas que estejam no seu próprio formato.
- O Microsoft Access, que foi a base de dados escolhida, por reunir o maior número de condições necessárias à execução do nosso trabalho, apresentar uma grande probabilidade de evolução, em termos de performance e ferramentas de desenvolvimento e futuras versões para outras plataformas (Macintosh).

[7] Integrated Database Aplication Programming Interface - Definição de uma API usada para aceder a bases de dados criada pela Borland, para uso nas suas bases de dados.

3.5 Escolha do sistema de informação geográfica

A escolha do sistema de informação geográfica a utilizar foi simplificada pelo facto de infelizmente, apenas existir um SIG adequado para o fim em vista. Aquilo que pretendemos é um SIG para Windows, com uma boa interface com o utilizador, programável e com capacidade de aceitar conversações DDE. Apenas o Mapinfo para Windows, suporta neste momento todas as facilidades que desejamos.

O Mapinfo para Windows da Mapinfo Corporation, é um dos SIG's mais utilizados para a iniciação e aprendizagem dos conceitos básicos de um SIG, devido à sua excelente interface com o utilizador. Suporta apenas informação geográfica do tipo vectorial, o que não é uma limitação no nosso caso. As facilidades de georeferenciação são excelentes, sendo apenas um pouco difíceis de utilizar uma vez que o fabricante supõe, que os mapas irão ser comprados, sendo depois corrigidos pelo utilizador. Esta suposição, válida para os clientes dos Estados Unidos, não se aplica de maneira nenhuma ao nosso caso.

A importação de ficheiros de informação geográfica é limitada, uma vez que apenas lê ficheiros DXF e de Mapinfo para Dos. Normalmente os SIG's estão preparados para fazer a importação de um número muito maior de tipos de ficheiros. Se quisermos usar outro tipo de ficheiros, temos de os traduzir para um formato intermédio chamado MIF (Mapinfo Interchange Format). As tabelas podem ser lidas do formato dBase, Lotus 123, Excel ou simples ficheiros ASCII com um caracter como separador. Uma das facilidades mais avançadas do Mapinfo é o facto de possuir um comando chamado SQL Select, que suporta condições geográficas, tais como Contains, Contains Entire, Within, Entirely Within e Intersects [Mapinfo 93a].

Para desenvolver uma aplicação em Mapinfo podemos usar o MapBasic que é uma linguagem estruturada, com um excelente suporte para a criação de aplicações típicas do Windows, com suporte para todos os tipos de dados que o Mapinfo usa, rotinas para a implementação de um cliente DDE e suporte para a chamada a DLL's externas [Mapinfo 93b].

- Uma das grandes qualidades do Mapinfo, é a existência de Mapinfo mais MapBasic para Macintosh e Unix (X-Windows), sendo o código do MapBasic

portável entre as várias plataformas. Segundo rumores recentes, a Intergraph escolheu-o para servir como visualizador, na plataforma Windows, do seu sistema SIG.

3.6 *Conclusão*

As escolhas por nós efectuadas, tomaram em consideração principalmente a expansão futura do nosso protótipo e a sua manutenção a longo prazo, assim como a facilidade e baixo custo da sua interligação com outros sistemas. Achamos que esses objectivos foram conseguidos apesar do lançamento futuro (anunciado a curto prazo) de novas plataformas, que devem assegurar um melhor desempenho. Mas essas plataformas, durante algum tempo, ainda devem manter um preço elevado (considerando a totalidade de um sistema e não apenas o preço de um computador), e pouco software disponível (em comparação com outras plataformas).

4. O ambiente de desenvolvimento

4.1 Introdução

Vamos neste capítulo apresentar as principais características do Microsoft Access, analisar as suas limitações, compará-lo com o seu maior concorrente para a produção de base de dados, que é o Visual Basic, e fazer um resumo sobre o Access Basic como linguagem de programação.

4.2 Características principais

As principais características do Access são uma excelente interface com o utilizador, um ambiente de desenvolvimento que permite minimizar o tempo de desenvolvimento de um protótipo e a existência de drivers ODBC[8] para quase todos os tipos de bases de dados, que permitem incorporar numa base de dados, tabelas de outros tipos de base de dados, mesmo que estam sejam remotas. A gama de drivers ODBC é vasta incluindo drivers para dBase, Clipper, Oracle, INGRES, Paradox, Sybase, DB2, SQL/400, SQL Server, Btrieve, DB2/DBM, INFORMIX, NetWare SQL, Excel, PROGRESS, SQLBase e XDB.

A existência de QBE (Query By Example) gráfico, permite que um utilizador faça interrogações sofisticadas à base de dados, com um mínimo de treino. E o QBE tem como opção a hipótese de criar código SQL a partir de uma interrogação, o que facilita o uso do SQL em Access Basic. O Access tem um preço razoável e, com o Access Distribution Kit, é possível criar executáveis que podem ser distribuídos e utilizados, sem o pagamento de licenças. Uma das características peculiares do Access é o facto das tabelas, indíces e aplicações de uma base de dados, fazerem parte de um único ficheiro, o que simplifica a instalação de aplicações.

A programação em Access é facilitada pela existência de dois níveis de programação: macros e Access Basic. Os macros em Access são simples listas de acções, enquanto que o Access Basic é uma linguagem de programação completa. Toda a programação é "event-driven", não tendo o programador que gerar código

[8] Open Database Connectivity - Definição de uma API usada para aceder a bases de dados criada pela Microsoft, para ser usada no acesso a base de dados relacionais, mesmo remotas.

para a construção da interface ou para a deteção de eventos. Apesar do Access não ser uma base de dados orientada aos objectos (OODBMS), possui um ambiente de desenvolvimento orientado aos objectos. As bases de dados em Access constituem uma "super-classe" que actua como "contentor" para todas as outras classes de objectos da base de dados. Os objectos tabela encapsulam as propriedades e os métodos, e essas propriedades e métodos são herdados por outros objectos tais como impressos, interrogações e relatórios, que usem as tabelas. Podemos ultrapassar as propriedades das tabelas, tais como valores por defeito e métodos, com propriedades atribuídas ao nível do impresso ou do relatório; isto vai criar uma forma elementar de polimorfismo. O "overloading" de funções em Access não é permitido, mas o tipo de dados Variant permite escrever código que atinge os mesmos objectivos.

O Access Basic herdou de versões antigas do Basic apenas o nome, sendo uma linguagem fortemente estruturada, muito aparentada com o Pascal, sendo "compilada" para uma espécie de P-code antes de ser executada, com uma compilação quase transparente para o utilizador. Como característica interessante para um programador, temos o facto do Access Basic ser um dialecto do Visual Basic for Applications (também conhecido por Object Basic), que a Microsoft pretende incorporar em quase todas as suas aplicações. Neste momento, o Excel 5.0 já incorpora o VBA, e prevê-se que a próxima versão do Word também o inclua. As vantagens de se ter apenas uma linguagem de customização de aplicações, sejam estas uma base de dados, uma folha de cálculo ou um processador de texto, são óbvias, porque permitem a um programador a reutilização de código e principalmente, evitam que este tenha que aprender uma linguagem por aplicação.

Outra característica poderosa do Acess Basic é o facto de permitir facilmente chamadas a DLL"[9]s quer estas sejam originais do Windows ou criadas para um fim específico. Apenas algumas rotinas do Windows que usam "callbacks"[10] são mais difíceis de utilizar.

[9] Dynamic Link Libraries - Livrarias dinâmicas de funções, usadas no ambiente Windows, que não suporta a "linkagem" estática de programas.

[10] Os "callbacks" são rotinas que um programador tem de escrever, e que são chamadas pelo sistema para o processamento de certos eventos. Para que isso aconteça, o programador tem de fornecer ao sistema um apontador para a rotina.

Os macros em Access representam um novo método de programação de aplicações. Normalmente, os macros de aplicações são gerados gravando um conjunto de teclas e escolhas de menus. Quando um macro é executado, ele simula as acções das teclas, dando origem aos comandos pretendidos. Em Access não podemos gravar macros; em vez disso, escolhemos as accções que queremos que o macro efectue entre as acções disponíveis numa "combo box". A maioria dos macros em Access necessita de um conjunto de qualificadores, chamados argumentos, para determinar os objectos a que a acção se aplica e o que a acção vai fazer.

Este tipo de macros, será usado no futuro em mais aplicações da Microsoft. Esta afirmação fundamenta-se num "white paper" publicado pela Microsoft no fim de 1991, em que se previa a criação de uma linguagem de macros comum (CML) para aplicações Windows, que permitisse a comunicação entre elas. O OLE 2.0 lançado em 1993, estabelece a arquitectura chamada automação OLE, segundo a qual a interacção entre as aplicações será realizada. O Microsoft Access é a primeira aplicação a possuir macros que seguem as normas da CML e é de esperar que outras aplicações lhe sigam o exemplo.

A incorporação de tipos de dados não-tradicionais em bases de dados, como imagens e objectos multimédia, é feita normalmente de duas maneiras: ou a base de dados suporta campos do tipo imagem, como o Superbase e o Paradox para Windows, ou suporta BLOB's (Binary Large Objects) que podem ser ficheiros de qualquer tipo. O Access adopta uma alternativa mais flexível, possuindo campos que podem ser objectos OLE. Assim um campo pode ser qualquer documento passível de ser criado por uma aplicação que tenha capacidade para ser um servidor OLE. Isto vem facilitar a tarefa da criação e manipulação de objectos multimédia, uma vez que o Access delega toda a responsabilidade nos programas adequados, o que permite fazer alterações no modo de armazenamento das imagens, por exemplo, sem tocar na estrutura da base de dados ou no seu código. A limitação mais vísivel do OLE manifesta-se no desempenho da base de dados, quando o computador em que esta corre não tem memória suficiente para o Access mais a aplicação servidora de OLE.

Se estivermos prontos a utilizar uma dose razoável de código em Access Basic, podemos simular BLOB's utilizando os campos OLE. É claro que utilizar os campos

OLE, sem ser para guardar objectos OLE, não é uma técnica aprovada pela Microsoft, mas podemos sempre recorrer a ela como última opção.

Em termos de DDE, o Microsoft Access pode ser tanto um cliente como um servidor, sendo as suas capacidades de servidor bastante poderosas, com um conjunto alargado de instruções. Pode-se, por exemplo, correr determinado macro do Access ou accionar uma interrogação escrita em SQL, que irá devolver os resultados via DDE ao programa cliente.

Outra das virtudes do Access, é o facto de ser um programa concebido desde o início para o funcionamento em rede, não existindo como noutras aplicações de bases de dados, versões para um único utilizador e versões para rede. O Microsoft Access incorpora de origem, características de segurança, que permitem a definição de utilizadores, grupos de utilizadores, e permitem definir permissões para leitura e edição da base de dados, assim como para a execução de interrogações e macros. Estas caracterísitsticas de segurança servem, normalmente, para proteger a base de dados de erros ocasionais e devem ser complementadas pelas facilidades de segurança do software da rede, uma vez que são pouco fiáveis.

4.3 *As suas limitações*

A documentação de aplicações escritas em Access é problemática, uma vez que o Access não possui rotinas incorporadas para a geração de um esquema da base de dados e dos dicionários de dados. Em sua substituição, o Access apenas possui uma livraria de funções, que lê as tabelas do sistema da base de dados, extrai a informação que as tabelas do sistema contêm e coloca-a numa base de dados. Esta livraria apenas cria tabelas, sendo da responsabilidade do utilizador a transformação destas em algo mais legível. A documentação de macros é um caso problemático, por isso muitos programadores substituem macros por código em Access Basic, uma vez que este pode ser copiado para ficheiros de texto, onde pode ser anotado e impresso.

Muitas das propriedades dos objectos nos impressos e relatórios não podem ser alteradas em "run-time". Propriedades, como as "captions"[11] de botões e etiquetas são fixas, não podendo ser alteradas seja com código de Access Basic ou com macros. Isto é uma grande limitação, normalmente contornada com soluções pouco elegantes.

O Access possui regras de validação de dados associadas às tabelas, que são verificadas automaticamente quando se faz a entrada dos valores ou a sua edição manualmente. Os valores de defeito, que são automaticamente atribuídos aos campos quando se cria um novo registo, não são verificados. Da mesma maneira, os registos que são alterados ou adicionados através de uma interrogação, também não são verificados. Para resolver este problema temos de criar um macro ou uma função em Access Basic, que complemente as regras de validação.

A integridade referencial entre tabelas não é assegurada, quando uma das tabelas não pertence ao Access. Normalmente, quando se definem relações por defeito entre tabelas, temos a opção de assegurar a integridade referencial ou não, nas relações que se criam. No entanto, as tabelas externas ao Access não fazem parte das tabelas entre as quais podemos definir relações em Access. A maioria dos servidores de SQL assegura a integridade referecial internamente, mas quando usamos tabelas em formato dBASE ou Paradox, precisamos de usar macros para a assegurar.

A criação de macros é fácil, mas a sua utilização pouco aconselhável, uma vez que na versão "run-time" do Access, um erro num macro tem como consequência o fim imediato da aplicação, sem qualquer aviso ou mensagem de erro. Só em Access Basic, conseguimos ter mecanismos de detecção e processamento de erros. Por esse motivo, depois da prototipagem de uma aplicação, a maioria dos macros são convertidos a Access Basic, para que se possam detectar e corrigir condições de erro, no seu funcionamento.

Existem ainda certas limitações no dialecto de SQL do Access, uma vez que não inclui os comandos CreateTable, DropTable, Subselect e Union.

[11] "Caption" é o nome dado em Access Basic ao texto colocado em botões, ou usado como legenda de janelas.

4.4 *O Visual Basic como alternativa*

Com o lançamento do Visual Basic 3.0, que também suporta drivers ODBC para acesso a bases de dados, o Access Engine para a criação de base de dados e uma maior flexibilidade na programação em comparação com o Access, a decisão de escolher esta plataforma e o Microsoft Access tornou-se confusa e difícil.

Devemos usar o Access quando necessitamos de:

- Desenvolver rapidamente um protótipo, uma vez que os métodos de desenho de impressos e os macros do Access, ajudam-nos a criar um protótipo num espaço de tempo reduzido.

- Desenvolver aplicações que os destinatários finais poderão modificar ou expandir. Muitas vezes os programadores de Access criam o núcleo de uma aplicação e deixam que os utilizadores finais criem ou "afinem" os impressos conforme as suas necessidades.

- Criar novas bases de dados ou mudar a estrutura de bases de dados existentes. Mesmo que no produto final usemos o Visual Basic, é mais fácil usar o Access para conceber e testar a base de dados.

- Criar livrarias de funções que que sejam usadas por várias aplicações em Access.

- Imprimir relatórios formatados dos dados criados com a aplicação. Note-se o Crystal Reports do VB 3.0 é independente do Visual Basic e necessita de tabelas como fonte de informação (além do pagamento de uma licença por cada cópia do Crystal Report Writer que se distribua).

- Usar o modo QBE (Query By Example) gráfico. A versão "run-time" do Access não permite usar este modo, mas permite que nas interrogações se usem funções definidas pelo programador, o que não acontece com o VB.

- Incluir ou ligar objectos OLE em campos da base de dados. O Visual Basic 3.0 não suporta campos com objectos OLE directamente, sem software adicional.

Deve-se usar o Visual Basic quando necessitamos de:

•Ter um controlo absoluto sobre o aspecto da aplicação, uma vez que em Access todos os impressos são filhos da janela principal do Access.

•Minimizar os recursos computacionais necessários. Quase todas as aplicações em Visual Basic se contentam com 4Mb de RAM, enquanto uma aplicação normal em Access necessita de 8Mb para correr razoavelmente.

•Passar SQL directamente ao servidor no caso de uma configuração cliente-servidor. O Access necessita para isso de um DDL que não é suportado pela Microsoft.

•Mudar as propriedades dos objectos em "run-time", uma vez que em Access temos de activar o modo de "design", para conseguir mudar as propriedades de um objecto e depois voltar ao modo normal para visualizar o objecto.

•Criar gráficos e inserir objectos dentro dos gráficos. Note-se que os gráficos criados com o VB são mais rápidos do que os criados com o Access.

•Processar gráficos ou outros tipos de dados não-tradicionais, armazenados em campos BLOB de tabelas de bases de dados cliente-servidor.

•Usar propriedades e eventos que estão disponíveis em Visual Basic, mas não existem em Access, ou não podem ser alterados em "run-time".

•Usar "controls" inteligentes que podem ser ligados directamente a tabelas da base de dados.

•Implementar numa aplicação as propriedades de um cliente OLE 2.0.

•Criar ou modificar base de dados e tabelas em "run-time", uma vez que o SQL do Access não inclui os comandos do SQL DDL (Data Definition Language). Como alternativa podemos usar os drivers ODBC para bases de dados do tipo "desktop", que têm capacidades de DDL (CREATE TABLE, CREATE INDEX), suportam os mesmos tipos de base de dados que o Access, e podem ser usados tanto pelo Access como pelo Visual Basic. Note-

se que se quisermos escrever uma ferramenta CASE para Access, é aconselhável que o façamos em Visual Basic e não no Access Basic.

Além de assinalarmos as diferenças entre as duas plataformas de desenvolvimento, vamos também chamar a atenção para aquilo as duas plataformas partilham:

- Os objectos de acesso a dados do Access e do Visual Basic são quase idênticos. Os "Recordsets" (os objectos Table, Dynaset e Snapshot) partilham as mesmas propriedades e métodos nos dois dialectos, sendo independentes da estrutura da base de dados e das suas tabelas.

- O Access Basic e o Visual Basic derivam da mesma linguagem, podendo-se transferir código nas duas direcções, com modificações de cerca de 20% a 25% do código, para que este funcione [Heinen 94]. No caso de código que faça chamadas à API do Windows, as declarações das funções e as suas chamadas em Visual Basic ou Access Basic são iguais (Note-se que a sintaxe do Visual Basic está mais próxima dos standards do Object Basic e do OLE 2.0, do que a sintaxe do Access Basic).

- A metodologia de desenho de impressos é similar, e os objectos comportam-se da mesma maneira nos dois ambientes.

- O Access usa a estrutura de ficheiros .MDB (Microsoft Database) e o Visual Basic está fortemente vocacionado para o uso de ficheiros .MDB, uma vez que o Access Database Engine está optimizado para esse tipo de ficheiros. Se a utilizarmos para a nossa base de dados teremos a opção de usar o Access ou o Visual Basic para as nossas aplicações, mantendo as mesmas tabelas de dados.

Para a nossa aplicação decidimos usar o Access, para minimizar o tempo de desenvolvimento, e poder usar campos OLE para as imagens. Esta decisão não é limitativa, uma vez que se no futuro tivermos alguma limitação de desempenho ou flexibilidade, podemos mudar a aplicação para Visual Basic, ou chamar DLL's externos escritos por exemplo em Visual C++, ou fazer as duas coisas. Neste momento, prevê-se que a futura versão 2.0 do Access incorpore a tecnologia de

acesso a índices "Rushmore", que faz parte do FoxPro, desaparecendo assim as limitações de velocidade do Access.

Uma das nossas preocupações fundamentais é a possibilidade de manutenção e preservação dos dados a longos prazo, uma vez que se tratam de documentos com interesse histórico. Mesmo que de futuro o Access seja abandonado, este permite que se exportem os dados para os formatos de texto delimitado, texto de largura fixa, Excel, Lotus 123, Paradox, dBASE, Microsoft SQL Server, Sybase e Oracle. O armazenamento das imagens em campos OLE parece ser, então, o principal factor a limitar a portabilidade da base de dados. Mas, com algumas linhas de código, numa das linguagens que existem para a "automação" do Windows, podemos exportar facimente as imagens, seja qual for o seu formato de destino, mesmo que nos campos OLE estam estejam armazenadas em formatos diferentes.

4.5 *A programação em Access Basic*

Um programador habitado a linguagens como o C e o Pascal, tem poucas dificuldades em compreender a sintaxe do Access Basic e programar nessa linguagem. O problema mais grave é a adaptação a um ambiente orientado aos objectos e dominado pelos eventos, como o ambiente de programação em Windows. A orientação aos objectos, do Access Basic pode ser discutível, mas não há dúvida que se trata de uma linguagem dominada por eventos. Isto porque não existe um programa, no sentido clássico do termo, sendo todo o código associado a eventos, que o programador não precisa de detectar.

Outra particulariedade do Access é o facto do código estar contido em módulos, que actuam como livrarias de funções e sub-procedimentos. Normalmente as funções e sub-procedimentos em Access Basic possuem um âmbito global, conduzindo a nomes únicos para cada função e sub-procedimento.

De seguida vamos descrever os três elementos de código em Access Basic que podemos usar num módulo.

4.5.1 A secção de declarações

Cada módulo de Access Basic, possui um secção de declarações que contém os elementos de código que se aplicam a todo o módulo, aparecendo nessa secção as seguintes instruções:

- Opções que se aplicam a todo o código nesse módulo e recebem o nome de variáveis de ambiente. A mais importante é a instrução **Option Explicit**, que obriga a que uma variável seja declarada antes do seu uso. A instrução **Option Compare** deixa-nos escolher entre comparações de strings sensíveis à diferença entre maiúsculas e minúsculas (**Binary**) ou não (**Text**). O Access suporta uma terceira opção (**Database**), para que nós possamos usar a ordenação normal de texto das bases de dados, quando fazemos comparação de strings.

- Declarações de variáveis e constantes usando as instruções **Dim VarName** e **Const ConstName**. As variáveis e constantes que declaramos na secção de declarações, são visíveis para todas as funções do módulo.

- Declarações de variáveis e constantes globais usando as instruções **Global VarName** e **Global Const ConstName**.

- Declarações dos protótipos das funções que fazem parte de DLL's. Estes protótipos são criados com as instruções **Declare Function** e **Declare Sub**, e são globais como as outras funções e sub-procedimentos do Access Basic.

A secção de declarações de um módulo de Access Basic, é muito similar em conteúdo e finalidade a uma "header" file da linguagem C ou C++. Note-se que o espaço de memória necessário, para as variáveis e constantes, só é alocado em "run-time".

4.5.2 Funções em Access Basic

As funções em Access Basic servem para devolver valores a expressões, e para criar métodos que respondem a eventos, que ocorrem quando corremos a nossa aplicação. O código que constitui a função, fica localizado entre as instruções **Function**

FuncionName() e **End Function**. Os argumentos da função são colocados dentro dos parênteses, e usamos vírgulas para os separar.

Exemplo:

```
'=============================================================
' A função seguinte converte um dígito entre 0 e 9 em texto
'=============================================================
  Function GetDigit (Digit)
     Select Case Val(Digit)
        Case 1: GetDigit = "Um"
        Case 2: GetDigit = "Dois"
        Case 3: GetDigit = "Três"
        Case 4: GetDigit = "Quatro"
        Case 5: GetDigit = "Cinco"
        Case 6: GetDigit = "Seis"
        Case 7: GetDigit = "Sete"
        Case 8: GetDigit = "Oito"
        Case 9: GetDigit = "Nove"

        Case Else: GetDigit = ""
     End Select
  End Function  ' Fim da função GetDigit
```

Note-se que em Visual Basic usamos sub-procedimentos para responder a eventos, enquanto que em Access Basic usamos funções.

4.5.3 Sub-procedimentos em Access Basic

Quando não precisamos de devolver um valor, ou fazer rotinas que serão atribuidas a eventos, podemos usar sub-procedimentos, para organizar o nosso código. O código de um sub-procedimento em Access Basic, encontra-se colocado entre as linhas **Sub Subname** e **Sub End**.

Exemplo:

```
'=============================================================
' Este sub-procedimento converte um dígito entre 0 e 9 em texto
'=============================================================
```

```
Sub  SubDigit (Digit,StrDigit)
   Select Case Val(Digit)
      Case 1: StrDigit = "One"       '
      Case 2: StrDigit = "Two"       '
      Case 3: StrDigit = "Three"     '
      Case 4: StrDigit = "Four"      ' Assign a numeric word value
      Case 5: StrDigit = "Five"      ' based on a single digit.
      Case 6: StrDigit = "Six"       '
      Case 7: StrDigit = "Seven"     '
      Case 8: StrDigit = "Eight"     '
      Case 9: StrDigit = "Nine"      '

      Case Else: StrDigit = ""       '
   End Select
End Sub  'End Sub SubDigit - return to calling program
```

Para chamar um sub-procedimento, podemos usar a instrução **Call Subproc (Param)**, ou muito simplesmente **Subproc Param**, separando os parâmetros do sub-procedimento por vírgulas, no primeiro caso, e por espaços, no segundo. A maioria dos programadores em Access, usa exclusivamente funções, e nos casos em a função não precisa de devolver um valor, esta devolve **True** se a sua execução não teve problemas e **False** se foi detectado algum erro.

4.5.4 Variáveis e a sua definição em Access Basic

Normalmente, as linguagens de programação compiladas, como o Pascal e o C exigem que cada variável seja declarada, antes do seu uso, definindo o seu nome e o seu tipo. Nas linguagens interpretadas, como o Basic Normal, e os dialectos do dBase, podemos atribuir um valor a uma variável, sem qualquer espécie de declaração prévia. O Access Basic que, permite que através do uso da linha Option Explicit na secção de declarações de um módulo, cada variável tenha de ser declarada antes do seu uso. Por defeito, o Access Basic atribui às variáveis não declaradas, o tipo Variant.

Existem várias vantagens ligadas à declaração das variáveis e dos seus tipos:

- O código escrito em Access Basic é mais rápido, quando usamos variáveis de outro tipo que não o tipo de dados Variant.

•Os erros causados pela escrita incorrecta do nome das variáveis desaparecem. Por defeito, quando encontra uma variável nova (nome mal escrito), o Access Basic cria uma nova variável do tipo Variant, "escondendo" assim erros de escrita.

•O código fonte fica mais legível e de compreensão mais fácil para outros (e para o próprio), quando declaramos as variáveis antecipadamente, sendo a linha onde declaramos uma variável, o sítio indicado para descrever o seu uso, com um comentário.

•Quando estiverem disponíveis compiladores para Access Basic, não necessitaremos de rever o código.

4.5.5 Tipos de dados existentes em Access Basic

O Access Basic possui uma variedade de tipos de dados mais rica do que a maioria das outras bases de dados existentes, para computadores pessoais. Na tabela seguinte, encontram-se os tipos de dados convencionais, que temos ao nosso dispor em Access Basic. (Os tipos de dados existentes nos campos da base de dados, estão relacionados com os tipos de dados disponíveis em Access Basic, apesar de serem distintos).

Tipo de Dados	*Tipo do Campo*	*Valor Mínimo*	*Valor Máximo*
Integer	*Byte, Integer, Yes/No*	*-32.768*	*32.767*
Long	*Counter, Long Integer*	*-2.147.483.648*	*2.147.483.647*
Single	*Single*	*-3,402 E+38* *1,401 E-45*	*-1,401 E-45* *3,402 E+38*
Double	*Double*	*-1,797 E+308* *4,940 E-324*	*-4,940 E-324* *1,797 E+308*
Currency	*Currency*	*-922.337.203.685.477,5 808*	*922.337.203.685.477,5 807*
String	*Text*	*0 caracteres*	*65.500 caracteres*

Tabela 1

Devido à importância das bases de dados, o Access Basic também possui outros tipos de dados (objectos), apropriados à utilização e manipulação de bases de dados.

Tipo de Objecto	*Objecto da Base de dados*
Database	*A base de dados na sua totalidade. Os objectos do tipo Database contêm todos os outros tipos descritos nesta tabela.*
Table	*Tabelas contidas num objecto Database (incluindo as tabelas lógicamente ligadas que se encontram noutros formatos).*
QueryDef	*A definição de uma interrogação a tabelas de um objecto Database, expressa em Access SQL*
Dynaset	*Uma tabela virtual (imagem em memória) de um objecto Table, criada do conjunto de resultados duma interrogação SQL (qualquer base de dados) ou de um objecto QueryDef (apenas o Access). Todos os objectos Dynaset baseados em tabelas, e alguns objectos em instruções SQL e em objectos QueryDef podem ser actualizados. Os objectos Dynaset são dinâmicos (daí o seu nome), de modo a reflectirem as mudanças inevitáveis dos dados quando se usa uma base de dados num ambiente multi-utilizador.*
SnapShot	*Uma tabela virtual equivalente à imagem fixa de um objecto Dynaset. Os SnapShot's não podem ser actualizados, e não reflectem as mudanças que ocorreram nos dados, depois que o Snapshot foi criado.*

Tabela 2

Quando criamos uma variável do tipo objecto, apenas criamos uma instância do objecto. Uma instância é uma referência ao objecto, na forma de um apontador, não uma cópia do objecto. Assim, podemos declarar várias variáveis que se referem ao mesmo objecto; sendo cada variável uma instância do objecto. Se, por exemplo, criarmos duas variáveis do tipo de objecto Table, usando a mesma tabela, podemos posicionar o apontador para registos de cada variável (instância) para um registo diferente.

A versão presente do Access Basic suporta 11 diferentes tipos de dados de campos, que conseguem abranger quase todos os tipos de dados existentes, em todos os SGBD's cliente-servidor e para computadores pessoais, exceptuando o tipo BLOB.

ID do tipo de campo	*Tipo de campo em Access Basic*	*Comentários e tipos de dados noutros SGBD's*
1	*Yes/No (Integer)*	*Campos lógicos em dBase; bit no SQL Server*
2	*Número (Byte)*	*Pode também representar um único char ASCII; tinyint no SQL Server*
3	*Número (Integer)*	*Short number em Paradox, inexistente em dBase; smallint no SQL Server*
4	*Número (Long)*	*Inexistente em Paradox e dBase; integer no SQL Server*
5	*Currency*	*Diferente do tipo currency do Paradox; money no SQL Server*
6	*Número (Single)*	*Float no SQL Server*
7	*Número (Double)*	*Campos numéricos em dBase e no Paradox; real no SQL Server*
8	*Data/Hora (formato da Microsoft)*	*Campos data em dBase e Paradox; datetime e smalldatetime no SQL Server*
9	*(reservado)*	
10	*Texto (String)*	*Campos de caracteres em dBase, campos alfanuméricos em Paradox; campos char e varchar do SQL Server*
11	*Objecto OLE*	*Pode também ser usado para armazenar BLOB's, tendo o seu tamnho limitado apenas pelo tamanho máximo de uma tabela.*
12	*Memo (String)*	*Campos memo do dBase e do Paradox; campos de texto do SQL Server, com a limitação máxima de 32000 caracteres.*

Tabela 3

Quando, os objectos relacionados com as bases de dados, apareceram pela primeira vez (no Visual Basic 1.0), estavam limitados pelas seguintes lacunas:

- Capacidade de utilização do valor Null, devolvido pelas bases de dados cliente-servidor, e por alguns SGBD's para computadores pessoais. O valor Null indica que não existem dados, no respectivo campo, de um registo pertencente a uma tabela.

•Um mecanismo para usar e criar indíces, em campos de tabelas, que possuam tipos diferentes, sem recorrer a conversões explícitas ("typecasting").

A Microsoft criou o tipo de dados Variant para resolver estes problemas, e para fazer a programação mais fácil para principantes, eliminando preocupações com a consistência dos vários tipos de dados. Na tabela 4 temos uma lista dos subtipos do tipo de dados Variant, assim como dos tipos de campos a que estão associados.

ID Subtipo Variant	*ID Tipo do Campo*	*Tipo de conteúdo da variável*
0	*(None)*	*Vazio (variável) por inicializar*
1	*(None)*	*Null (desconhecido, não existem dados válidos)*
2	*3,1,2*	*Integer (2 bytes); os tipos de campos Logical e Byte também são convertidos para Variant's do subtipo 2*
3	*4*	*Long Integer (4 bytes)*
4	*5*	*Single (número em vírgula flutuante de precisão simples, 4 bytes)*
5	*6*	*Double (número em vírgula flutuante de precisão dupla, 8 bytes)*
6	*7*	*Currency (número em vírgula fixa, com quatro casas decimais)*
7	*8*	*Data/Hora (no formato próprio da Microsoft)*
8	*10,11,12*	*String (string de caracteres covencional); os campos memo e OLE também se identificam como Variant's do subtipo 8.*

Tabela 4

O tipo de dados Variant fornece as vantagens do polimorfismo, sem a necessidade do "overloading" de funções, uma vez que o ID do subtipo serve para o Access Basic fazer as conversões necessárias em "run-time", tentando sempre fazer com que o resultado uma expressão com Variant's seja um Variant, que tenha um número de subtipo, do menor valor que seja possível.

Podemos ainda em Access Basic definir novos tipos de dados, resultantes da associação de tipos de dados convencionais (equivalentes às estruturas da linguagem C), e matrizes com um número de dimensões menor que 60, e um limite de 32767 elementos. O limite ao número de elementos das matrizes pode ser ultrapassado

através do uso de "huge arrays" que podem ir até 64 Mbytes ou usando tabelas em vez de matrizes.

4.6 Conclusão

O Microsoft Access contitui um ambiente de trabalho poderoso, e de fácil utilização. A aprendizagem do Access Basic é interessante, estando as suas capacidades muito além dos BASIC's com que nós contactamos durante a licenciatura. O seu potencial é enorme, uma vez que constitui um dialecto do Visual Basic for Applications, que será incorporado de futuro na maioria das aplicações da Microsoft.

O Visual Basic pode ser considerado um competidor do Access, mas apenas em primeira aproximação, porque numa análise mais aturada, conseguimos ver que os dois ambientes de desnvolvimento se complementam.

Estamos confiantes que a maioria das limitações do Access desaparecerão na sua versão seguinte, pelo menos as suas limitações em relação ao Visual Basic, porque essas existem por razões comerciais e não tecnológicas

5. Implementação do Protótipo

5.1 Introdução

Neste capítulo pretendemos descrever as facetas que julgamos relevantes da implementação do nosso protótipo. Iremos analisar as estruturas de dados utilizadas, a concepção da interface com o utilizador, as regras usadas na programação em Access Basic, e iremos explicar alguns detalhes significativos da programação do nosso protótipo

5.2 Definição da estrutura de dados

A estrutura de dados foi sendo redefinida e refinada à medida que a nossa compreensão do projecto foi aumentando. Inicialmente apenas consideramos duas entidades: prédios e azulejos, podendo cada prédio possuir vários azulejos (Figura 11).

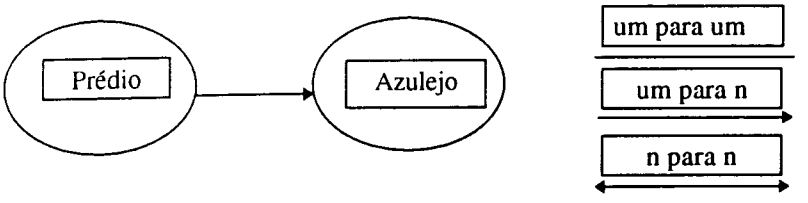

Figura 11 Primeira aproximação

Como um azulejo pode existir em mais do que um prédio, temos de separar os dados de um azulejo comuns aos vários prédios onde ele se encontra aplicado, como a cor e o tamanho, dos dados referentes à existência concreta de um azulejo num prédio como a quantidade de azulejos, o seu estado de conservação, etc. Esta separação conduziu-nos ao passo seguinte na evolução da estrutura (Figura 12).

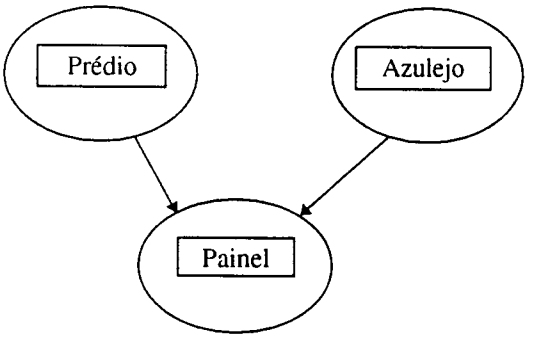

Figura 12 Primeiro refinamento

Para a inclusão das imagens, consideramos que normalmente as imagens são produzidas para documentar os paineis, conduzindo esse raciocínio a uma nova estrutura (Figura 13).

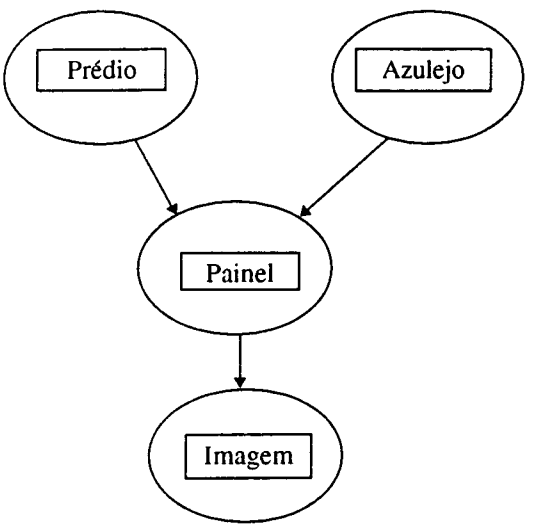

Figura 13 Inclusão das imagens

Esta estrutura simples complica-se com as relações entre cada azulejo e as suas cores, que é uma relação de n para n, uma vez que cada azulejo pode possuir várias cores, e cada cor pode ser utilizada em muitos azulejos (Figura 14).

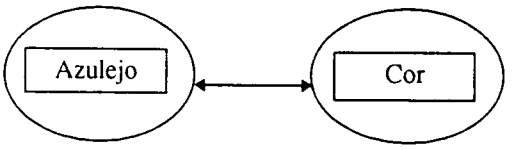

Figura 14 Relação entre azulejos e cores

Esta relação de n em n foi normalizada, com o recurso a uma tabela intermédia (Figura 15).

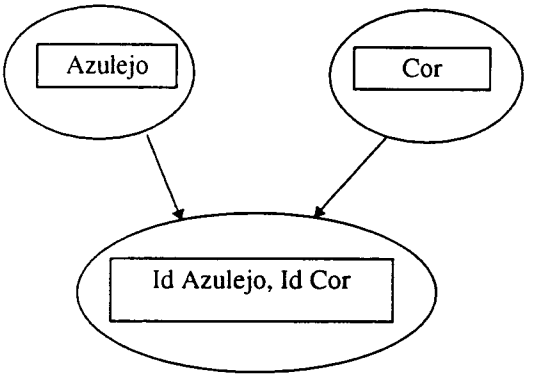

Figura 15 Normalização da relação entre azulejos e cores

O diagrama final da estrutura da nossa base de dados será obtido com a inclusão da relação anterior (Figura 16).

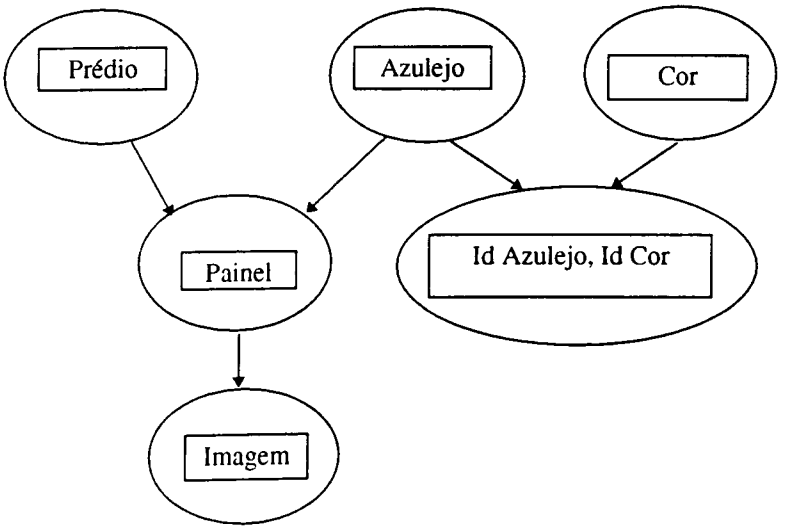

Figura 16 Diagrama final da estrutura da base de dados

A tabela correspondente aos prédios, com a informação respectiva, deverá em sistemas futuros, ser uma tabela remota, incorporada não na base de dados, mas num SIG, de preferência no SIG central da Câmara.

Se a tabela dos prédios, estiver incorporada no SIG central da Câmara, é provável que tenha campos adicionais, e as vantagens dessa incorporação serão as seguintes:

- A introdução dos dados relativos a cada prédio (dessa tabela), deixará de estar a cargo da divisão de Museus e Património Histórico e Artístico, poupando trabalho.
- As eventuais discrepâncias dos dados entre o SIG da Câmara e a base de dados, ficam totalmente eliminadas.
- A informação disponível no SIG da Câmara, sobre todos os prédios, poderá ser utilizada pela divisão, para as suas tarefas.
- O SIG da Câmara poderá funcionar como veículo para a troca de informação, entre os diferentes serviços da Câmara.

Outras tabelas foram incorporadas, nesta estrutura, mas com o objectivo de satisfazer plenamente as necessidades dos utilizadores e a funcionalidade do programa e não completar a base de dados.

Nome da Tabela	*Nome do Campo*	*Tipo do Campo*	*Compr. do Campo*	*Comentário*
Azulejos	*ID azulejo*	*Long*	*4*	*Chave primária*
Azulejos	*Assunto*	*Text*	*20*	
Azulejos	*Tamanho Vertical*	*Double*	*8*	*Ou mais correctamente, dimensão num sentido e noutro*
Azulejos	*Tamanho Horizontal*	*Double*	*8*	
Azulejos	*Tamanho do Quadrado*	*Double*	*8*	*Tamanho do quadrado, em número de azulejos, válido apenas para azulejos de padrão*
Azulejos	*Subtipo*	*Text*	*20*	*Liso, Padrão, Relevo, Figura Avulsa e Figurativo*

Nome da Tabela	*Nome do Campo*	*Tipo do Campo*	*Compr. do Campo*	*Comentário*
Azulejos	*Tipo*	*Text*	*20*	*Azulejo avulso, painel, telha, mosaico, painel cerâmico, ou painel de pastilha*
Azulejos	*Método de Manufactura*	*Text*	*15*	
Azulejos	*Descrição*	*Text*	*20*	
Azulejos	*Autor da Ficha*	*Text*	*20*	
Azulejos	*Observações*	*Memo*	*0*	
Cores	*ID Cor*	*Long*	*4*	*Chave primária*
Cores	*Nome da Cor*	*Text*	*20*	
Cores	*Valor da Cor*	*Long*	*4*	*32 bits da cor no formato Windows*
Cores dos Paineis	*ID Objecto*	*Long*	*4*	
Cores dos Paineis	*ID Cor*	*Long*	*4*	
Cores dos Paineis	*ID Tinta*	*Long*	*4*	*Chave primária*
Épocas	*NomedaÉpoca*	*Text*	*20*	*Chave primária*
Épocas	*DataInf*	*Double*	*8*	*Data inferior de uma época*
Épocas	*DataSup*	*Double*	*8*	*Data superior de uma época*
Imagens	*ID Imagem*	*Long*	*4*	*Chave primária*
Imagens	*ID Painel*	*Long*	*4*	*ID Painel*
Imagens	*Tipo da Imagem*	*Text*	*15*	*Curta descrição do tipo da imagem*
Imagens	*Colocação*	*Text*	*15*	*Colocação da imagem no arquivo (físico)*
Imagens	*Imagem*	*OLE Object*	*0*	*A imagem (mesmo)*
Paineis	*ID Painel*	*Long*	*4*	*Chave primária*
Paineis	*Número Total de Azulejos*	*Double*	*8*	
Paineis	*Época*	*Text*	*20*	*Texto ou Data*
Paineis	*Data Inferior*	*Double*	*8*	*Data colocada no campo época ou data inferior correspondente*
Paineis	*Data Superior*	*Double*	*8*	*Data colocada no campo época ou data superior correspondente*
Paineis	*Origem do Fabrico*	*Text*	*20*	*Fábrica, oficina, atelier, ou zona geográfica*
Paineis	*Autor*	*Text*	*20*	
Paineis	*Marcas*	*Text*	*20*	*Marcas existentes, na parte posterior dos azulejos*
Paineis	*Estado de Conservação*	*Text*	*8*	*Bom, razoável, ou mau*
Paineis	*Avaliação*	*Text*	*20*	

Nome da Tabela	*Nome do Campo*	*Tipo do Campo*	*Compr. do Campo*	*Comentário*
Paineis	*Data de Incorporação*	*Date/Time*	*8*	
Paineis	*Autor da Ficha*	*Text*	*20*	
Paineis	*Colocação*	*Text*	*15*	*Colocação no armazém*
Paineis	*Observações*	*Memo*	*0*	
Paineis	*ID Prédio*	*Long*	*4*	*ID Prédio*
Paineis	*Exterior*	*Yes/No*	*1*	*Azulejo usado no exterior do edifício, ou no seu interior*
Paineis	*ID Azulejo*	*Long*	*4*	
Prédios	*ID Prédio*	*Long*	*4*	*Chave primária*
Prédios	*Nome*	*Text*	*30*	
Prédios	*Rua*	*Text*	*30*	
Prédios	*Número de Polícia*	*Double*	*8*	
Prédios	*Freguesia*	*Text*	*15*	*Freguesia (uma das 15 do Porto)*
Prédios	*Proprietário*	*Text*	*50*	
Prédios	*Morada do Proprietário*	*Text*	*40*	
Prédios	*Localização*	*Text*	*10*	*Esquerdo, Direito, A, B, etc..*

Tabela 5

5.3 *A interface com o utilizador*

O desenho de uma boa interface com o utilizador constitui uma das áreas mais difíceis da criação de um programa. Isto porque além dos problemas associados à complexidade normal da criação de software, as interfaces com o utilizador sofrem dos seguintes problemas [Myers 94]:

- Os programadores têm sérias dificuldades, em pensar como utilizadores normais de um programa. Deste modo, constroem normalmente as interfaces, tomando como utilizador normal, um utilizador que possua os seus conhecimentos sobre o programa, o seu funcionamento, os seus comandos e a utilização de computadores. Note-se que isto muitas vezes é feito pelos programadores inconscientemente, e sem nenhuma culpa, uma vez que ninguém consegue eliminar da memória conhecimentos préviamente adquiridos.

•O campo de actividades e conhecimentos necessários para o desenho de uma boa interface é vasto e complexo, passando pela programação, psicologia, design gráfico, escrita de documentos técnicos, etc.

•É imprescindível, para o desenho de uma interface com o utilizador, um conhecimento profundo da área de aplicação do programa. Esse conhecimento, tem uma apreensão normalmente difícil e demorada.

•As teorias e as regras que existem sobre a produção de interfaces com o utilizador são insuficientes para a produção de uma boa interface.

•O desenho de uma interface é um processo criativo, logo com uma formalização difícil, se não for impossível.

•O desenho da interface é feito acompanhando normalmente o método iterativo de construção de software, que não é o método mais adequado ao desenho de interfaces.

•O utilizador é normalmente bastante sensível ao desempenho da interface em termos de velocidade, exigindo uma resposta instantânea às suas acções.

•A programação de uma interface com o utilizador, é feita hoje em dia, fazendo apenas rotinas de resposta a eventos. A ferramenta utilizada para a construção de interfaces, tem normalmente a responsabilidade de detectar os eventos e fazer a sua gestão. Assim, os conhecimentos de estruturação de código, desenho de programas, e engenharia de software, ensinados normalmente sobre linguagens de programação tradicionais, não se aplicam aos nóvos métodos de criação de interfaces.

•A enorme quantidade de rotinas de resposta a eventos, existente num programa normal, torna difícil o seu "debugging", uma vez que todas essas rotinas são independentes, tornando as suas interacções difíceis de prever e controlar.

•A avaliação correcta de uma interface é um processo difícil, caro e lento.

Na elaboração deste sistema tentámos moldar a interface com o utilizador, às necessidades e expectativas do utilizador, de maneira a sugerir ao utilizador, as acções

a tomar em cada momento. Tentámos dar a impressão que o nosso programa nunca força o utilizador a decisões finais e irremediáveis, para que este nunca tenha a impressão, que está apenas a fazer aquilo que o computador ordena. Além de tentar usar botões e designações adequadas ao contexto do programa, tentámos que as "pistas" fornecidas pela interface fossem claras, não para o programador (que o são sempre), mas para o utilizador normal [Williams 92]. Tentámos também que o programa fosse utilizável, sem manuais, por experimentação pura e simples, e que o programa obedecesse na medida do possível, aos standards normais de aspecto e interface com o utilizador do Windows. [Microsoft 93b] Note-se que o simples facto de se seguir um standard para a interface com o utilizador, vem facilitar a familiarização com o programa por parte do utilizador, e o trabalho de desenho de uma interface por parte do programador, uma vez que este é normalmente uma pessoa pouco habilitada em termos de desenho de interfaces [Rettig 92].

O facto de usarmos nomes em vez de bitmaps nos botões, tem a ver com a grande dificuldade que é conseguir um desenho, que exprima uma acção de uma maneira clara e inequívoca, para qualquer tipo de utilizador. É claro que uma interface com icones é muito mais atractiva do ponto de vista estético, mas normalmente um utilizador tem alguma dificuldade inicial em saber o que é que cada icon faz. Por essa razão, nos programas mais recentes para o Windows, quando o cursor passa sobre um icon, aparece um texto descritivo da acção, numa caixa de texto, ou na linha de estado do programa. Muitos dos icones que aceitamos hoje em dia como normais, são aceites por habituação e não por razões de lógica. Por exemplo, nas aplicações da Microsoft: se abrir um ficheiro corresponde a abrir uma pasta, porque é que guardar um ficheiro não corresponde a fechar uma pasta, mas sim a uma diskette, sabendo nós que o ficheiro normalmente não vai ser guardado em diskette?

Este problema manifesta-se quando uma interface com o utilizador usa uma metáfora que o utilizador não está em condições de comprender. Como exemplo, podemos apontar o "reciclador" do NextStep, que na altura da sua criação, não era um símbolo de compreensão fácil para um utlizador português típico, que não estava familiarizado com tal símbolo.

Para uma maior flexibilidade da interface com o utilizador, procuramos dividir a interface, no maior número possível de janelas independentes, de forma permitir a cada utilizador uma disposição das janelas adequada às suas necessidade e preferências.

5.4 Regras usadas na programação

Na programação em Access Basic e noutras linguagens de programação que admitem um grande número de tipos de dados, um programador comete normalmente o erro de atribuir a uma variável um valor do tipo errado, ou esquecer-se do tipo de uma variável. Para alguém que não se encontre familiarizado com o código, é necessária uma leitura constante das declarações de variáveis, para que se possa entender o código.

Na elaboração dos nosso código usamos um conjunto de regras que definem um estilo de criação de nomes de variáveis que permite uma maior legibilidade do código. Este método de "baptismo" de variáveis chama-se Húngaro, devido à nacionalidade do seu criador, Charles Simonyi [Symonyi 91]. Alguns elementos deste estilo de programação são usados pela Microsoft nos manuais do Visual Basic, e nos manuais do SDK (Software Development Kit) do Windows. Internamente, a Microsoft tambem o usa, assim como muitos programadores das mais variadas origens, que o usam para tornar o seu código mais legível.

Na sua versão mais simples a regras especificam uma série de identificadores que devem iniciar o nome de uma variável, identificando o seu tipo. Uma grande parte dos programadores que trabalham com o Access, adaptou essas convenções da maneira seguinte (Tabela 6, Tabela 7 e Tabela 8)[ACCESS-L 94].

Como exemplo do uso destas convenções temos a seguinte declaração: **Dim curSalary As Currency** .

Tabela 6

Tipo de campo	*Identificador*
Binary	*bin*
Byte	*byt*
Counter	*lng*

Tipo de campo	*Identificador*
Currency	*cur*
Date/Time	*dtm*
Double	*dbl*
Integer	*int*
Long	*lng*
Memo	*mem*
OLE	*ole*
Single	*sng*
Text	*str*
Yes/No	*ysn*

Tabela 7

Tipo de Objecto	*Identificador*
Chart	*cht*
Check box	*chk*
Combo box	*cbo*
Command button	*cmd*
Frame	*fra*
Label	*lbl*
Line	*lin*
List box	*lst*
Option button	*opt*
Option group	*grp*
Rectangle (shape)	*shp*
Subform/report	*sub*
Textbox	*txt*
Toggle button	*tgl*

Tabela 8

Tipo de Variável	*Identificador*
Control	*ctl*
Currency	*cur*
Database	*db*

Tipo de Variável	*Identificador*
Double	*dbl*
Dynaset	*dyn*
Flag	*f*
Form	*frm*
Integer	*int*
Long	*lng*
QueryDef	*qry*
Report	*rpt*
Single	*sng*
Snapshot	*snp*
String	*str*
Table	*tbl*
Variant	*var*
Yes/No	*ysn*

Este método que pode parecer estranho numa primeira aproximação, constitui a nosso ver uma excelente ferramenta de apoio à documentação de um programa.

5.5 Detalhes de implementação

Vamos de seguida descrever os métodos utilizados para a implementação de algumas facilidades do programa, que nós julgamos importantes, e chamar a atenção para detalhes, que sendo de uma importância relativa podem constituir obstáculos importantes na programação em Access Basic.

5.5.1 Funcionalidades especiais

Na elaboração do nosso sistema, certas funcionalidades especiais, que apareceram por sugestão dos utilizadores, não são usuais numa base de dados normal.

Uma dessas necessidades é a possibilidade da introdução da data de um painel, como um número ou como uma época, por ex: 1870, ou Séc XIX, ou 2ª metade do séc XVIII. Para que as interrogações à base de dados, funcionem correctamente

devemos converter as épocas em números. Por outro lado, para que a época introduzida pelo utilizador, não seja destruida, esta também deve ser guardada. Para converter a época em limites superiores e inferiores, usámos uma tabela adicional, com o nome de cada época, e os seus limites inferiores e superiores. Na tabela dos paineis, além de um campo do tipo texto, para a data, temos dois campos numéricos para os limites inferiores e superiores da data.

Quando se processa a entrada de dados, o programa vê se o texto da data pode ser convertido num número, se for esse o caso, esse número é colocado nos campos dos limites superiores e inferiores da data. Se o texto da data corresponder a uma época, as datas correspondentes são pesquisadas na tabela das épocas, e colocadas na tabela dos painéis. Esta solução pouco elegante, e que aumenta a redundância e o tamanho da base de dados, não é tão má como parece á primeira vista. Isto porque o tamanho dos campos adicionais é reduzido, uma vez que se tratam de números inteiros de um valor baixo, e o processamento de interrogações que envolvam a época fica simplicado, com o uso dos limites superiores e inferiores da data de cada painel.

Outra das características do programa que foi sugerida pelos utilizadores é a ajuda que se encontra disponível, para a identificação das cores usada em cada azulejo. No modo de identificação de cores, quando seleccionamos um ponto de uma imagem, o programa vai procurar na lista de cores catalogadas, uma lista das mais próximas, de modo a que a tarefa do utilizador na escolha das cores esteja facilitada. Este método de classificação de cores permite também evitar desvios próprios do "scanner" e do monitor utilizados, se estes se mantiverem mais ou menos inalterados, e construirmos um catálogo inicial das cores classificando-as com base nos espécimens originais.

A identificação das cores mais próximas é feita a partir do cálculo da distância geométrica de uma cor a outra, considerando que a posição de uma cor no espaço é dada pelas suas componentes RGB.

5.5.2 Obstáculos encontrados

Encontrámos algumas dificuldades imprevistas na realização deste projecto, devido ao comportamento do Access nalgumas situações.

A validação dos dados é uma área problemática da construção de uma base de dados em Access, porque os eventos a que normalmente um programador associa a validação dos dados, não cobrem todas as hipóteses possíveis de erro. Como exemplo: além da validação individual de cada campo, devemos detectar o fecho de uma janela, não o permitindo se algum dos campos não possui valores correctos. Para obter este tipo de segurança não devemos apenas detectar o fecho de uma janela, depois da janela se fechar, mas antes da janela se fechar.

As facilidades oferecidas pela existência do valor Null, para assinalar a inexistência de dados válidos, podem-se tornar uma origem de erros, quando o programador não prevê que esse valor possa surgir nos dados. Esta detecção deve ser efectuada quase sempre, porque por definição de Null, qualquer expressão que envolva esse valor tem como resultado Null.

A comunicação com o Mapinfo por DDE revelou-se de implementação um pouco difícil, devido aos problemas normais que surjem sempre que se tenta fazer o "debugging" da comunicação entre dois programas, devido ao Mapinfo não aceitar comandos por DDE, apenas leitura e modificação de variáveis. Assim necessitamos de fazer algum código em MapBasic, para realizar um mínimo de integração. Quando algo não funcionava, tinhamos dois possíveis culpados: o nosso código em Access Basic e o nosso código em MapBasic. Apesar das dificuldades, consideramos o mecanismo de DDE muito útil e simples para a comunicação entre dois programas.

Os mecanismos de georeferenciação existentes no Mapinfo, possuem uma utilização directa difícil, uma vez que a unidade de georeferenciação recomendada em Mapinfo é o "segmento de rua". O método de armazenamento da informação relativa às ruas é o seguinte:

- A cada rua corresponde um identificador, existindo uma tabela com duas entradas: nome da rua e id_rua.

- Cada rua pode ter vários segmentos, estando os segmentos armazenados numa tabela de que constam o identificador da rua correspondente, a informação geométrica do segmento, e os números de polícia correspondentes

às casas colocadas nos extremos do segmento (tanto do lado esquerdo como do direito).

Assim quando necessitamos de georeferenciar um prédio em Mapinfo este obtém primeiro o identificador correspondente à rua, indo depois percorrer a tabela dos segmentos, para identificar o segmento em que o prédio se encontra. A posição do prédio é obtida a partir do número de polícia, por interpolação simples entre os extremos do segmento de rua, produzindo as coordenadas do prédio.

Este método de armazenamento de informação complica a operação inversa, de obtenção dos endereços a partir da selecção geométrica no ecrã, uma vez que ou criamos entidades correspondentes a todos os prédios, georeferenciando todos os prédios, ou usamos algoritmos de "line-clipping" para determinar a lista de endereços correspondente a uma selecção geométrica. Felizmente este problema só se põe, se quisermos obter todos os endereços possíveis, a partir de uma selecção geométrica, e no nosso caso, apenas desejamos obter a lista dos endereços correpondentes aos prédio já georeferenciados.

5.6 Conclusão

Tentámos implementar o protótipo assegurando a simplicidade da estrutura de dados, o máximo de funcionalidade, uma interface com o utilizador razoável, e a legibilidade do código. Isto para que todos os seus desenvolvimentos futuros sejam fáceis e feitos com segurança.

6. Conclusões

6.1 Introdução

Neste capítulo tentamos fazer um resumo das conclusões a que chegámos e perspectivar futuros desenvolvimentos e possibilidades de expansão para o nosso protótipo.

6.2 Resultados obtidos

O nosso sistema encontra-se ainda em fase de teste e afinação, mas achamos que os resultados obtidos até agora são positivos, tendo em conta as alterações constantes que o sistema tem sofrido, e o interesse manifestado pela divisão de Museus e Património Histórico e Artítico da Câmara Municipal do Porto na sua utilização.

As decisões que tomamos de início revelaram-se acertadas, uma vez que os pormenores mais importantes do sistema, que são a plataforma e a estrutura da informação, têm permanecido inalteráveis, enquanto a interface com o utilizador, e a funcionalidade do protótipo têm sofrido alterações consecutivas, à medida que redefinimos a funcionalidade do sistema e os nossos conhecimentos sobre azulejos aumentam.

O Microsoft Access revelou-se uma plataforma excelente para a prototipagem de aplicações, mas tem algumas limitações, quando queremos um controlo absoluto e total sobre a nossa aplicação. O balanço global é positivo, devido ao curto prazo de desenvolvimento de uma aplicação, que se consegue, permitindo assim percorrer mais vezes o ciclo de desenvolvimento de software (especificação, design, implementação, debugging e análise) no mesmo espaço de tempo.

A interface com o SIG, é muito interessante uma vez que permite revelar informação que não pode ser visualizada de nenhum outro modo e colocar questões a que uma base de dados convencional não responderia.

6.3 *Desenvolvimentos futuros*

A transformação do nosso protótipo, numa aplicação multiposto é facílima, porque não envolve nenhuma modificação ao software, sendo apenas aconselhável a definição de utilizadores da base de dados. As facilidades do Access relativamente ao seu funcionamento em rede são um auxiliar precioso nessa transição.

A digitalização das imagens será, no futuro, incorporada na base de dados, através do comando de um programa apropriado ao scanner ou scanners envolvidos nessa acção. A automação de um programa já existente é mais fácil do que a escrita de software de acesso directo ao scanner, uma vez que se o scanner mudar, os princípios de automação de outro programa serão os mesmos, enquanto que a programação do scanner será bem diferente.

Encontra-se em fase de especificação e estudo uma aplicação que permita uma maior integração da base de dados com o SIG, apresentando ao utilizador apenas uma interface. No futuro, pensamos integrar o nosso protótipo com o SIG que irá ser adoptado pela Câmara do Porto. Esse será o nosso maior desafio, uma vez que envolverá o uso de tabelas remotas e fornecerá uma integração perfeita entre as divisões da Câmara Municipal que estão directa ou indirectamente envolvidas com a preservação do património histórico e artístico da cidade.

Num prazo de tempo mais curto, prevemos a expansão dos assuntos da base de dados, para outros objectos e motivos de interesse, que necessitam de ser inventariados e documentados, para que possam ser protegidos e preservados. Esta expansão será fácil, uma vez que uma das partes mais complexas deste projecto, que é a interligação entre a base de dados e o sistema de informação geográfica permanecerá inalterável.

6.4 *Conclusão*

Estamos seguros que o nosso sistema vai ainda sofrer algumas alterações, na sua funcionalidade e interface, e que mesmo que a interface, o sistema, ou a plataforma se alterem, os dados permanecerão inalterados e conseguirão sobreviver a todas as alterações.

Cremos que, desse modo, um dos principais objectivos do nosso protótipo, que é assegurar a conservação dos dados, foi atingido. Possuímos assim, uma base sólida a partir da qual poderemos partir para programas mais poderosos e funcionais.

7. Bibliografia

[ACCESS-L 94]
ACCESS-L Mailing List
Janeiro de 1994

[Andrews 93]
Dave Andrews and Selinda Chiquoine
"Moving Toward Windows Building Blocks"
BYTE December 1993

[Bearman 94]
David Bearman
"Conferência sobre a Informatização de Museus"
Casa TAIT 31 Janeiro 1994

[Berst 94]
Jesse Berst
"Caution: OLE 2.0 Ahead"
Visual Basic Programmers Journal *February/March 1994*

[Brierly 93]
Dean Brierly
"Photo Conservation: Archival Storage and Presentation"
Camera & Darkroom May 1993

[Britton 93]
David Britton
Comunicação Privada
Conferência Multimédia 93

[Brocskschmidt 93]
Kraig Brocskschmidt
"Programming for Windows with Object Linking and Embedding 2.0"
Preliminary Draft 1993

[Browning 93]
Dave Browning
"Relational Databases: New Blood, New Power"
PC Magazine May 11 1993

[Butler 93]
Brian Butler and Sheryl Canter
"PC-Based SQL: Time to Commit?"
PC Magazine October 12 1993

[Canter 93]
Sheryl Canter
"SQL: Putting Up a Good Front"
PC Magazine November 9 1993

[DeVoney 93a]
Chris DeVoney
"Rough Start on The Road to NT"
Windows Sources July 1993

[DeVoney 93b]
Chris DeVoney & Randall C. Kennedy
"NT Has Arrived"
Windows Sources November 1993

[Heinen 94]
Roger Heinen
"Searching for a Unified Theory"
Visual Basic Programmers Journal *February/March 1994*

[Kennedy 93]
Randall C. Kennedy
"NT Server Widens Windows Reach"
Windows Sources July 1993

[Kennedy 94]
Randall C. Kennedy
"Unlock the Power of OLE"
Windows Sources March 1994

[Kiyooka 94]
Gen Kiyooka
"Subclassing in OLE 2.0"
BYTE January 1994

[Mapinfo 93a]
Mapinfo Corporation
"Mapinfo Desktop Mapping Software Reference" 1993

[Mapinfo 93b]
Mapinfo Corporation
"MapBasic Desktop Mapping Tools Reference" 1993

[Microsoft 92]
Microsoft Press
"Object Linking and Embedding, Programmers Reference Version 1"
1992

[Microsoft 93a]
Microsoft
"Visual Basic Programmer´s Guide"
1993

[Microsoft 93b]
Microsoft
"Visual Design Guide"
1993

[Microsoft 94]
Microsoft Corporation
"Chicago Questions and Answers"
January 1994

[Miller 94]
Michael J. Miller
"Getting it Together"
PC Magazine February 8 1994

[Museum-L 94]
Artigos da "mailing list"MUSEUM-L
Fevereiro e Março de 1994

[Myers 94]
Brad A. Myers
"Challenges of HCI Design and Implementation"
interactions January 1994

[Nielsen 90]
Jakob Nielsen
"HyperText & HyperMedia"
Academic Press 1990

[Parsaye 89]
Kamran Parsaye, Mark Chignell, Setrag Khoshafian, Harry Wong
"Inteligent Databases"
Wiley 1989

[Pleas 93]
Keith R. Pleas
"Hands-On OLE"
Visual Basic Programmers Journal
December/January 1993/1994

[Rettig 92]
Marc Rettig
"Interface Design When You Don't Know How"
Communications of The ACM Vol 35 Nº1 January 1992

[Stephens 94]
Paul Stephens
"Talking Windows"
PCPLUS February 1994

[Symonyi 91]
Charles Symonyi and Martin Heller
"The Hungarian Revolution" *BYTE August 1991*

[Udell 94a]
Jon Udell
"Windows for Workgroups 3.11"
BYTE February 1994

[Udell 94b]
Jon Udell
"Chicago: An Ambitious Compromise"
BYTE March 1994

[Ullman 89]
Jeffrey D. Ullman
"Principles of Database Systems; Second Edition"
Computer Science Press 1989

[Watterson 94]
Karen Watterson
"Data on Demand"
Windows Sources February 1994

[Williams 92]
Thomas R. Williams and David R. Farkas
"Minimalism Reconsidered: Should We Design Documentation For Exploratory Learning"
SIGCHI Bulletin Vol 24 Number 2 April 1992