



COMPLÉMENT AU GUIDE COMPLET

Dessiner ses écrans et ses états **PDF**

Les concepteurs de l'atelier wdlexis+ : formulaires et états, dessinés à la souris puis animés en quelques lignes de Lexis+. Pas à pas, avec des exemples complets et commentés, exécutés pour de vrai.

2

concepteurs

31

contrôles dans la palette

11

zones et éléments d'état

5

exemples exécutés



Formulaires



États PDF



Référence



Aides-mémoires

Bienvenue dans les concepteurs

Ce complément prolonge le **Guide complet du débutant**. Le guide vous a appris à construire une fenêtre **par le code** (chapitre 13 : `interface.fenetre`, `:ajouter`, les gestionnaires `surclie` ...) et à écrire des fichiers. Il est temps de passer à la vitesse supérieure : l'atelier `wdlexis+` contient deux **concepteurs** qui dessinent pour vous.

- Le **concepteur de formulaires** : vous posez à la souris les contrôles d'une fenêtre (saisies, listes, boutons, onglets...), vous réglez leurs propriétés et vous choisissez les procédures qui répondent à leurs événements. Le concepteur écrit le code de la fenêtre dans un fichier `form_<nom>.prg`.
- Le **concepteur d'états** (ou éditeur d'états) : vous dessinez au millimètre des **zones horizontales** (en-tête, lignes, totaux, pied de page) avec des textes, des cadres, des traits et des images. Le programme les remplit avec le paquet `rapport`, qui produit un **PDF** paginé.

Comme dans le guide, chaque notion est illustrée par un **exemple complet**. Les programmes ont été **réellement exécutés** avec le runtime `wdlexisrt` pendant la fabrication de ce complément ; les captures d'écran viennent de l'atelier et des programmes eux-mêmes, et les pages de PDF reproduites ont été produites par le paquet `rapport`.

• Comment lire ce complément

- Les **onglets de couleur** sur le bord droit indiquent la partie : **indigo** pour les formulaires, **vert d'eau** pour les états, **bleu nuit** pour la référence, **ardoise** pour les aides-mémoires.
- Les encadrés sont ceux du guide : **À retenir** (l'essentiel), **Pour bien comprendre** (une notion expliquée simplement), **Astuce**, **Attention** (un piège fréquent), **Syntaxe** (la forme exacte d'une instruction) et **À vous de jouer** (un exercice).
- Dans le code, les **mots-clés**, les **chaînes**, les **nombres**, les **paquets**, les **@références** et les **commentaires** ont chacun leur couleur, comme dans l'atelier. Le badge **GÉNÉRÉ** signale un code écrit par un concepteur : on le lit, on ne le modifie pas.
- Refaites les manipulations dans l'atelier en même temps que vous lisez : c'est en dessinant qu'on apprend à dessiner.

• Ce qu'il faut savoir avant de commencer

- Les concepteurs travaillent dans un **projet** (un dossier) : ouvrez-en un avec *Fichier > Ouvrir un projet...* ou créez-le avec *Fichier > Nouveau projet...* (guide, section 1.5).
- Tous les fichiers `.prg` d'un projet sont chargés ensemble : les fichiers produits par les concepteurs n'ont pas besoin d'être importés (guide, section 4.7).
- Les fenêtres, contrôles et événements sont ceux du paquet `interface` (guide, chapitre 13).

• Les annexes

Annexe A : la référence complète des deux concepteurs (palette, propriétés, événements, paquet `rapport` et briques `interface.concepteur` / `interface.etat`). **Annexe B** : deux aides-mémoires d'une page, à imprimer et à garder sous la main.

Deux concepteurs, un même principe

Les deux concepteurs se ressemblent volontairement : une **palette** à gauche, le **dessin** au centre, la **hiérarchie** et l'**inspecteur** à droite, une barre d'outils en haut. Ce que vous apprenez avec l'un vous sert avec l'autre.

Formulaires

Fenêtres de saisie et de consultation : contrôles, dispositions, propriétés, événements reliés à vos procédures.

Chapitre 1

États PDF

Factures, listes, inventaires : zones d'en-tête, de pied et de détail, variables, pagination automatique.

Chapitre 2

	Concepteur de formulaires	Concepteur d'états
On dessine	une fenêtre et ses contrôles	des zones horizontales et leurs éléments
Unité	le pixel (ou la disposition)	le millimètre
Fichier produit	<code>form_<nom>.prg</code>	<code>etat_<nom>.prg</code>
Ce qu'il contient	la fonction <code>FORM<NOM>_OUVRIR()</code> et le dessin	la fonction <code>ETAT_<NOM>()</code> qui décrit l'état
Votre code	les procédures des événements	les valeurs des <code>{variables}</code> de chaque zone
Résultat	une fenêtre qui réagit	un fichier PDF paginé
Menu	<i>Conception > Nouveau formulaire...</i>	<i>Conception > Nouvel état...</i>

Pour bien comprendre

Dessin et code ne se mélangent pas. Le concepteur réécrit son fichier à chaque modification : vous ne touchez jamais à `form_<nom>.prg` ni à `etat_<nom>.prg`. Votre code vit dans vos propres fichiers, et il appelle ce que le concepteur a produit.

Astuce

Les deux ateliers ont les mêmes concepteurs : **wdlexis+** (l'atelier natif) et l'**atelier écrit en Lexis+** (`exemples/atelier`). Un formulaire ou un état dessiné dans l'un s'ouvre dans l'autre : les fichiers ont exactement le même format.

Sommaire

	1. Dessiner des fenêtres à la souris	5
1.1	Pourquoi un concepteur ?	5
1.2	Ouvrir le concepteur	6
1.3	La fenêtre du concepteur	7
1.4	La palette	8
1.5	Placer, déplacer, redimensionner	8
1.6	Les propriétés	9
1.7	Les événements	11
1.8	Le code produit par le concepteur	12
1.9	Exemple complet : la fiche client	14
1.10	Vos propres contrôles dans la palette	16
1.11	Le concepteur dans l'atelier en Lexis+	17
	2. Produire des états PDF	20
2.1	Qu'est-ce qu'un état ?	20
2.2	Ouvrir l'éditeur d'états	21
2.3	La fenêtre de l'éditeur	21
2.4	Les zones	22
2.5	Les éléments	22
2.6	Les variables	24
2.7	Le fichier produit par l'éditeur	25
2.8	Remplir l'état : le paquet rapport	25
2.9	Exemple complet : une facture de deux pages	26
2.10	Exemple complet : un inventaire par rayon	28
2.11	Vérifier : l'aperçu PDF et le code inséré	30
2.12	L'éditeur d'états dans l'atelier en Lexis+	31
	Annexe A · Référence	34
	Annexe B · Aides-mémoires	41

Dessiner des fenêtres à la souris

Posez les contrôles, réglez leurs propriétés, branchez leurs événements : le concepteur écrit le code de la fenêtre, vous n'écrivez que ce qui doit se passer.



- 1.1 Pourquoi un concepteur ?
- 1.2 Ouvrir le concepteur
- 1.3 La fenêtre du concepteur
- 1.4 La palette
- 1.5 Placer et déplacer
- 1.6 Les propriétés
- 1.7 Les événements
- 1.8 Le code produit
- 1.9 La fiche client
- 1.10 Contrôles personnalisés
- 1.11 Dans l'atelier en Lexis+

Dans ce chapitre, vous apprendrez à...

- ✓ ouvrir le concepteur et vous repérer dans sa fenêtre
- ✓ régler les propriétés d'un contrôle et de la fenêtre
- ✓ lire le fichier `form_<nom>.prg` produit
- ✓ ajouter vos propres contrôles à la palette
- ✓ choisir une disposition et y ranger les contrôles
- ✓ relier un événement à une procédure du projet
- ✓ ouvrir le formulaire et lire ses valeurs dans le programme
- ✓ construire un concepteur avec `interface.concepteur`

1.1 Pourquoi un concepteur ?

Au chapitre 13 du guide, vous avez construit des fenêtres **par le code** : on crée la fenêtre, on lui ajoute des conteneurs, puis des contrôles, un par un. C'est parfait pour trois boutons. Pour une fiche client de vingt contrôles rangés dans une grille, le code devient long, et surtout on ne **voit** le résultat qu'en exécutant le programme.



EXEMPLE Une fiche construite à la main

principal.prg

```
1 fen = interface.fenetre{titre = "Fiche client", largeur = 560, hauteur = 480, disposition =
  "bordure"}
2 identite = fen:ajouter(interface.cadre("Identité", "grille"))
3 identite.ancrage = "centre"
4 identite:ajouter(interface.etiquette("Nom"))
5 nom = identite:ajouter(interface.saisie{indication = "Dupont"})
```



EXEMPLE

Une fiche construite à la main (suite)

principal.prg

```
6 identite:ajouter(interface.etiquette("Prénom"))
7 prenom = identite:ajouter(interface.saisie{indication = "Marie"})
8 && ... et ainsi de suite pour chacun des 20 contrôles de la fiche
```

Le concepteur de formulaires renverse la démarche: vous **voyez** la fenêtre pendant que vous la construisez, avec les vrais contrôles de l'interface, et c'est lui qui écrit ces lignes. Il ne vous reste qu'à écrire ce que fait le programme quand l'utilisateur clique, saisit ou choisit.



Le vocabulaire du concepteur

- **Formulaire**: une fenêtre dessinée dans le concepteur, enregistrée dans `form_<nom>.prg`.
- **Contrôle**: un élément de la fenêtre (étiquette, saisie, bouton, liste...). Chacun a un **nom**, unique dans le formulaire.
- **Conteneur**: un contrôle qui en contient d'autres (cadre, panneau, onglets...); sa **disposition** décide comment ils sont rangés.
- **Propriété**: un réglage d'un contrôle (son libellé, sa couleur, sa largeur...).
- **Événement**: ce qui arrive à un contrôle (un clic, un changement de valeur...), relié à une **procédure** de votre code.

1.2

Ouvrir le concepteur

Le concepteur s'ouvre depuis le menu **Conception** de l'atelier, une fois un projet ouvert (sans projet, l'atelier vous demande d'abord d'en ouvrir un).

Commande	Effet
Conception > Nouveau formulaire pour le fichier courant...	crée un formulaire neuf; le fichier <code>.prg</code> affiché dans l'éditeur (à défaut, le fichier principal) recevra les procédures de ses événements
Conception > Ouvrir un formulaire...	rouvre un <code>form_<nom>.prg</code> existant, avec le fichier de code qui lui est associé
double-clic sur <code>form_<nom>.prg</code>	dans l'arbre du projet: même effet que la commande précédente

Un formulaire neuf s'appelle `1`, puis `2`... (le premier numéro libre): renommez-le tout de suite dans l'inspecteur (propriété **nom** de la fenêtre), par exemple `client`. Le fichier devient `form_client.prg`, et le concepteur suit ce changement dans votre code.

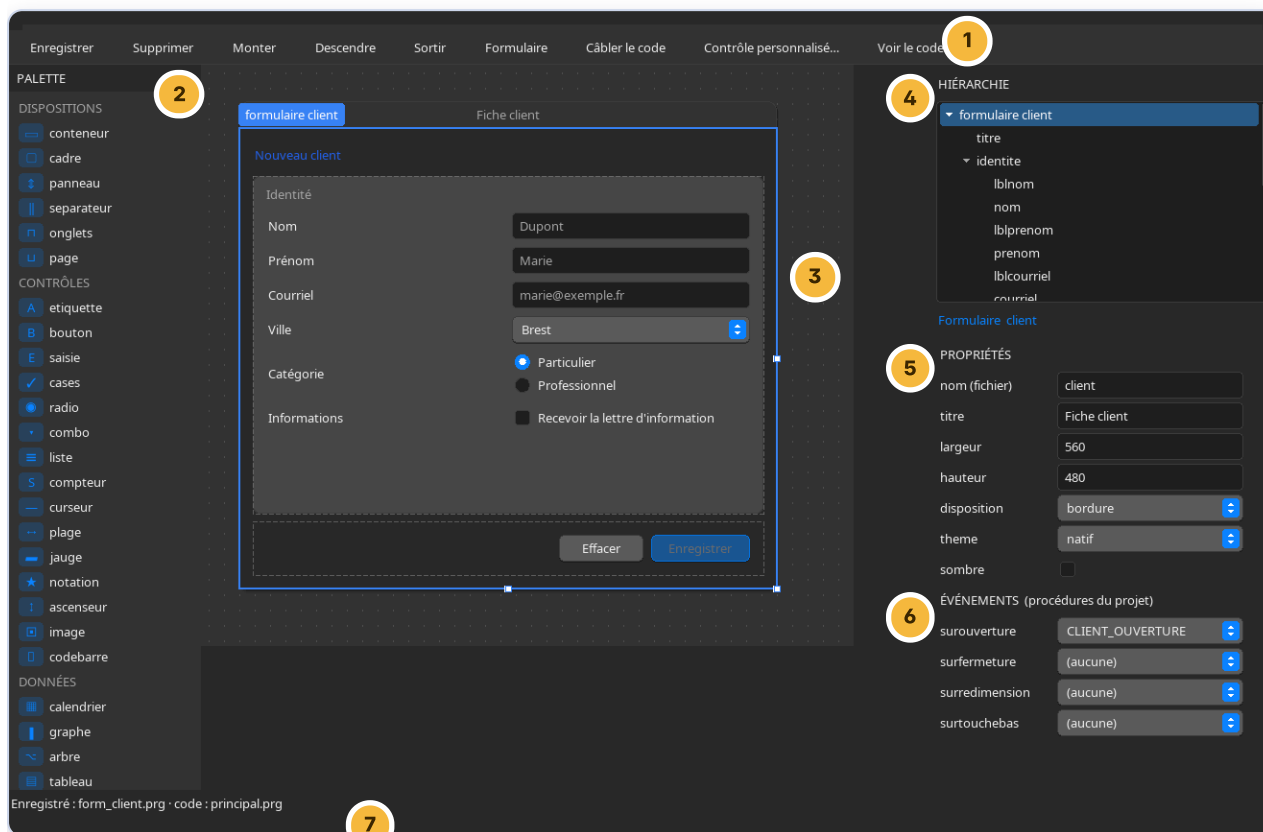


À retenir

Un formulaire est toujours relié à un **fichier de code**: celui qui reçoit ses procédures d'événements. Choisissez-le en affichant le bon fichier dans l'éditeur avant *Nouveau formulaire pour le fichier courant...*

1.3 La fenêtre du concepteur

Voici le concepteur ouvert sur la fiche client de l'exemple `exemples/formulaire`. La fenêtre du formulaire est sélectionnée : l'inspecteur montre ses propriétés et ses événements.



- 1 Barre d'outils** : enregistrer, supprimer, ordonner, câbler le code...
- 2 Palette** : les contrôles, par catégorie
- 3 Canevas** : le formulaire, dessiné avec les vrais contrôles
- 4 Hiérarchie** : l'arbre des contrôles
- 5 Propriétés** de la sélection, modifiables sur place
- 6 Événements** et procédures associées
- 7 Barre d'état** : fichiers enregistrés, avertissements

• La barre d'outils

Bouton	Effet
Enregistrer	régénère <code>form_<nom>.prg</code> (le concepteur le fait aussi à chaque modification)
Supprimer	supprime le contrôle sélectionné (touche Suppr)
Monter / Descendre	plus tôt ou plus tard dans l'ordre d'affichage ; en disposition absolue, l'ordre de superposition
Sortir	sort le contrôle de son conteneur
Formulaire	sélectionne la fenêtre elle-même (ses propriétés et ses événements)
Câbler le code	ajoute au code le bloc des procédures et, dans le programme principal, l'appel qui ouvre le formulaire (1.8)



Bouton	Effet
Contrôle personnalisé...	ajoute à la palette un contrôle créé par une fonction Lexis+ du projet (1.10)
Voir le code	affiche dans l'atelier le fichier de code associé

1.4 La palette

La palette présente les contrôles du paquet `interface` sous leurs noms Lexis+, rangés en trois catégories.

Catégorie	Contrôles
Dispositions	<code>conteneur</code> , <code>cadre</code> , <code>panneau</code> , <code>separateur</code> , <code>onglets</code> , <code>page</code>
Contrôles	<code>etiquette</code> , <code>bouton</code> , <code>saisie</code> , <code>cases</code> , <code>radio</code> , <code>combo</code> , <code>liste</code> , <code>compteur</code> , <code>curseur</code> , <code>plage</code> , <code>jauge</code> , <code>notation</code> , <code>ascenseur</code> , <code>image</code> , <code>codebarre</code>
Données	<code>calendrier</code> , <code>graphe</code> , <code>arbre</code> , <code>tableau</code> , <code>grille</code> , <code>kanban</code> , <code>mosaïque</code> , <code>gant</code> , <code>markdown</code> , <code>barremenu</code>

Pour ajouter un contrôle, deux gestes au choix :

- **glissez-le** de la palette jusqu'à l'endroit voulu du formulaire ;
- **cliquez-le** : il s'ajoute à la fin du conteneur sélectionné.

Chaque contrôle reçoit un nom automatique formé d'un préfixe et d'un numéro : `btn1` pour un bouton, `lbl1` pour une étiquette, `sail` pour une saisie, `cbo1` pour un combo... Donnez-lui un nom parlant (`nom`, `enregistrer`) : c'est par ce nom que votre code l'atteindra.

 **Astuce**

Des onglets commencent avec une page. Pour en ajouter une autre, sélectionnez les onglets et cliquez sur `page` dans la palette ; un clic sur l'en-tête d'une page l'affiche, et les contrôles ajoutés vont dans la page affichée.

1.5 Placer, déplacer, redimensionner

Dans une fenêtre wdgui, on ne place pas chaque contrôle au pixel près : on choisit une **disposition** pour chaque conteneur (et pour la fenêtre), et elle range ses enfants. C'est ce qui permet à la fenêtre de rester belle quand on l'agrandit.

Disposition	Rangement des enfants
vertical	les uns sous les autres
horizontal	côte à côte, de gauche à droite
grille	en tableau, ligne par ligne ; la propriété <code>colonnes</code> donne le nombre de colonnes (idéal pour « libellé, saisie »)
flux	à la suite, avec retour à la ligne quand la place manque

Disposition	Rangement des enfants
bordure	chaque enfant s'accroche en haut, en bas, à gauche, à droite ou au centre (propriété ancrage)
pile	les uns sur les autres, un seul visible à la fois
absolue	à la position exacte donnée par x et y

● À la souris

- Cliquez sur un contrôle pour le sélectionner ; ses **poignées** le redimensionnent, et celles de la fenêtre changent sa taille.
- Glissez un contrôle pour le déplacer : dans une disposition verticale, horizontale, grille ou flux, une **barre bleue** montre où il sera inséré ; dans une disposition bordure, la zone en surbrillance donne son ancrage ; lâché sur un autre conteneur, il y entre ; en disposition absolue, il se place librement.
- Des **repères magenta** apparaissent pendant un déplacement ou un redimensionnement : bords et centres alignés sur les autres contrôles et sur le conteneur, largeur ou hauteur égale à celle d'un autre contrôle (« = btn1 »), proportions du conteneur (« 1/4 », « 1/3 », « 1/2 », « 2/3 », « 3/4 », « tout »). Une pastille affiche la position ou la taille.

● Au clavier

Touche	Effet
Suppr	supprime le contrôle sélectionné
Échap	sélectionne le conteneur parent
flèches	en disposition absolue : déplace de 1 pixel
Maj + flèches	en disposition absolue : déplace de 10 pixels



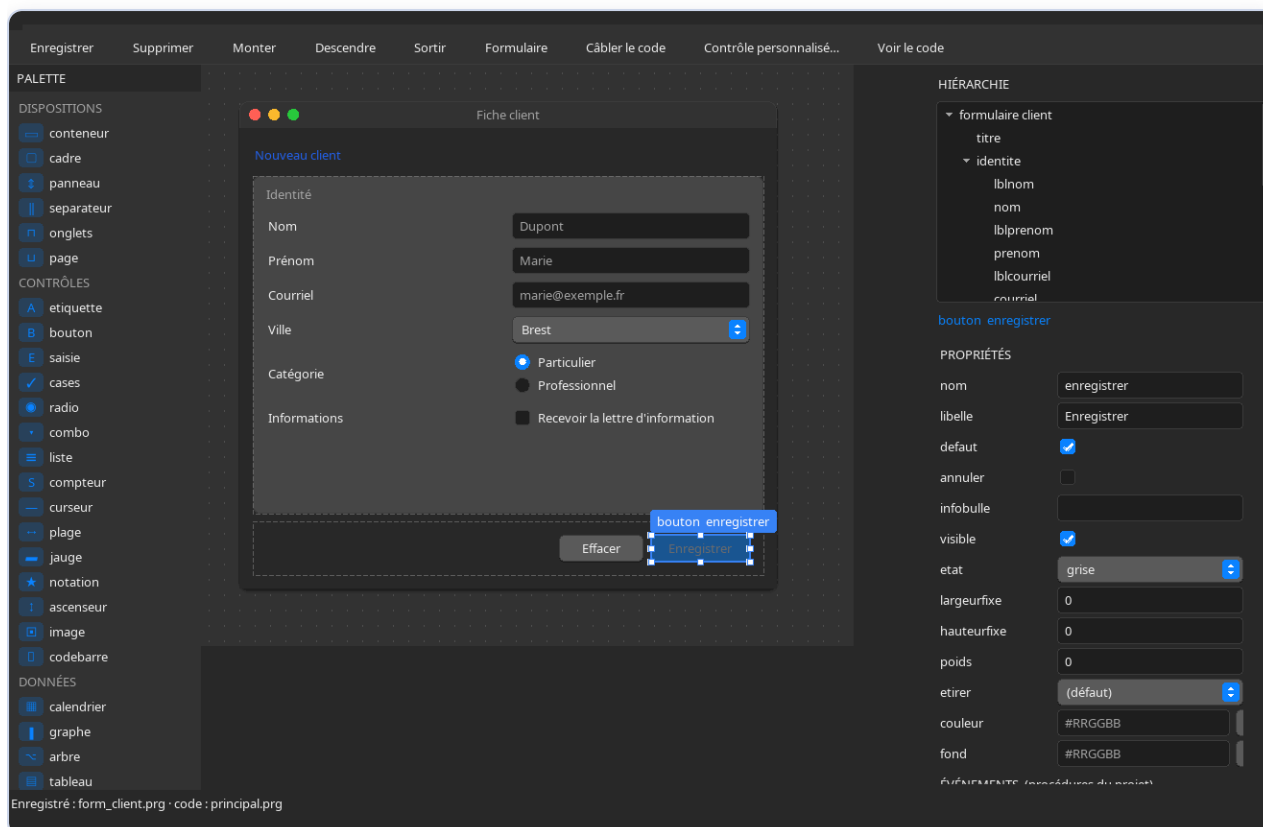
À retenir

Construisez de l'extérieur vers l'intérieur : la disposition de la fenêtre d'abord (souvent **bordure**), puis les grands conteneurs (un **cadre** en **grille** pour les champs, un **conteneur** **horizontal** pour les boutons), enfin les contrôles.

1.6

Les propriétés

L'inspecteur affiche les propriétés de la sélection, avec leurs noms Lexis+, et les modifie sur place : un champ de texte, une case à cocher, une liste de choix ou un sélecteur de couleur selon la propriété. Voici le bouton **enregistrer** sélectionné : sa propriété **etat** vaut **grise**, il sera donc désactivé à l'ouverture.



• Les propriétés de tous les contrôles

Propriété	Rôle
nom	le nom du contrôle dans le code ; s'applique par Entrée (ou en quittant le champ)
infobulle	le texte affiché au survol
visible	contrôle affiché ou caché
etat	actif , grise (désactivé) ou lecture (lecture seule)
largeurfixe, hauteurfixe	taille imposée (0 : taille naturelle)
x, y	position, en disposition absolue
poids	part de la place restante que le contrôle reçoit
etirer	s'étire ou non dans la largeur (vide : comportement par défaut)
ancrage	en disposition bordure : haut , bas , gauche , droite , centre
couleur, fond	couleur du texte et du fond (#RRGGBB , avec un sélecteur)

• Les propriétés qui contiennent une liste

Plusieurs propriétés décrivent une liste dans un seul champ, avec un point-virgule entre les éléments.

Propriété (contrôle)	Exemple
options (cases , radio)	Particulier;Professionnel
elements (combo , liste)	Brest;Nantes;Paris;Rennes
colonnes (tableau)	Nom;Valeur
colonnes (grille)	nom:Nom;prix: Prix unitaire (nom de la colonne : titre)
noeuds (arbre)	Racine;Racine/Enfant
menus (barremenu)	Fichier/Ouvrir;Fichier/Quitter;Aide/À propos
listes (kanban)	À faire;En cours;Fait

• Les propriétés de la fenêtre

Cliquez sur **Formulaire** pour sélectionner la fenêtre: **nom** (le nom du formulaire, donc du fichier), **titre** , **largeur** , **hauteur** , **disposition** , **theme** (**natif** , **windows** , **macos** , **linux**) et **sombre** .

1.7 Les événements

Sous les propriétés, l'inspecteur liste les **événements** du contrôle sélectionné. Chacun a une liste de choix :

- **(aucune)** : pas de procédure, l'événement est ignoré ;
- une **procédure du projet** à un paramètre (**procedure NOM(ev)**) : elle est associée ;
- **+ nouvelle procédure...** : le concepteur écrit la procédure dans le code associé, l'y associe et place le curseur dedans. Il la nomme d'après le contrôle et l'événement : **ENREGISTRER_CLIC** pour le **surcllic** du bouton **enregistrer** (avec un numéro si ce nom est déjà pris).

Contrôles	Événements
tous	surcllic , surdoublecllic , surcllicdroit , surentree , sursortie , surfocuse , surfinfocus , surtouchebas
saisie	en plus : surchangement , sursaisie
cases, radio, compteur, curseur, plage, notation, ascenseur, calendrier, separateur, onglets	en plus : surchangement
combo, liste, tableau, kanban, markdown, barremenu	en plus : surselection
arbre	en plus : surselection , surdeploiement , surrepli
grille	en plus : surselection , surchangement , survalidation
fenêtre	surouverture , surfermeture , surredimension , surtouchebas

**Attention**

Une procédure associée mais introuvable dans le projet (supprimée, renommée...) est signalée dans la barre d'état du concepteur. À l'exécution, l'association @NOM vers une procédure absente provoquerait une erreur : corrigez-la dans l'inspecteur.

1.8 Le code produit par le concepteur

À chaque modification, le concepteur réécrit `form_<nom>.prg`, à la racine du projet. Ce fichier ne contient qu'une fonction, comme tout fichier secondaire d'un projet, suivie du dessin lui-même, en commentaires `&&$` que le concepteur relit.

```

GÉNÉRÉ La fonction d'ouverture de la fiche client form_client.prg

1 && Formulaire « client » – généré par le concepteur de wdllexis+.
2 && Régénéré à chaque modification dans le concepteur : modifiez le dessin là-bas, pas ce
  fichier.
3 && FORMCLIENT_OUVRIIR() ouvre la fenêtre et renvoie ses contrôles, aussi rangés dans la
4 && variable globale formclient (formclient.fenetre, formclient.titre...).
5
6 fonction FORMCLIENT_OUVRIIR()
7   globale formclient
8   locale f = {}
9   f.fenetre = interface.fenetre{titre = "Fiche client", largeur = 560, hauteur = 480,
disposition = "bordure"}
10  f.titre = f.fenetre:ajouter(interface.etiquette("Nouveau client"))
11  f.titre.ancrage = "haut"
12  f.titre.couleur = "#2563EB"
13  f.identite = f.fenetre:ajouter(interface.cadre("Identité", "grille"))
14  f.identite.ancrage = "centre"
15  f.lblnom = f.identite:ajouter(interface.etiquette("Nom"))
16  f.nom = f.identite:ajouter(interface.saisie{indication = "Dupont"})
17  f.lblprenom = f.identite:ajouter(interface.etiquette("Prénom"))
18  f.prenom = f.identite:ajouter(interface.saisie{indication = "Marie"})
19  f.lblcourriel = f.identite:ajouter(interface.etiquette("Courriel"))
20  f.courriel = f.identite:ajouter(interface.saisie{indication = "marie@exemple.fr"})
21  f.lblville = f.identite:ajouter(interface.etiquette("Ville"))
22  f.ville = f.identite:ajouter(interface.combo("Brest", "Nantes", "Paris", "Rennes"))
23  f.lblcategorie = f.identite:ajouter(interface.etiquette("Catégorie"))
24  f.categorie = f.identite:ajouter(interface.radio("Particulier", "Professionnel"))
25  f.lblinfos = f.identite:ajouter(interface.etiquette("Informations"))
26  f.lettre = f.identite:ajouter(interface.cases("Recevoir la lettre d'information"))
27  f.boutons = f.fenetre:ajouter(interface.conteneur("horizontal"))
28  f.boutons.ancrage = "bas"
29  f.etat = f.boutons:ajouter(interface.etiquette(""))
30  f.etat.poids = 1
31  f.effacer = f.boutons:ajouter(interface.bouton("Effacer"))
32  f.enregistrer = f.boutons:ajouter(interface.bouton("Enregistrer", {default = vrai}))
33  f.enregistrer.etat = "grise"
34  && événements (procédures du code : principal.prg)
35  f.fenetre.surouverture = @CLIENT_OUVERTURE
36  f.nom.surchangement = @CLIENT_SAISIE
37  f.effacer.surclic = @CLIENT_EFFACER
38  f.enregistrer.surclic = @CLIENT_ENREGISTRER

```

GÉNÉRÉ

La fonction d'ouverture de la fiche client (suite)

form_client.prg

```

39   formclient = f
40   retourner f
41   finfonction
42
43   && Dessin du formulaire (relu par le concepteur) : ne pas modifier.
44   && {
45   &&   "kind": "fenetre",
46   &&   "name": "client",
47   &&   "props": {
48   &&     "titre": "Fiche client",
49   &&     "largeur": "560",
50   &&     "hauteur": "480",
51   &&   ...

```

Retenez trois choses de ce fichier :

- la fonction s'appelle `FORM<NOM>_OUVRIR()` : ici `FORMCLIENT_OUVRIR()` ;
- elle range chaque contrôle sous son nom dans une table, également placée dans la variable globale `form<nom> (formclient) : formclient.nom, formclient.enregistrer ...` ;
- les événements y sont associés aux procédures de votre code (`f.enregistrer.surcllic = @CLIENT_ENREGISTRER`).

• Le bloc des procédures dans votre code

Les procédures créées par **+ nouvelle procédure...** sont rangées à la fin du code associé, dans un bloc délimité par deux lignes de marque. Le concepteur s'en sert pour retrouver l'endroit où écrire : gardez-les.

GÉNÉRÉ

Le bloc marqué, juste après une nouvelle procédure

principal.prg

```

1   && ---- formulaire « client » : début (généralisé par le concepteur, gardez ces marques) ----
2   && FORMCLIENT_OUVRIR() ouvre le formulaire ; ses contrôles sont dans formclient
   (formclient.fenetre...).
3
4   procedure ENREGISTRER_CLIC(ev)
5     && surcllic de enregistrer (formulaire « client »)
6
7   finprocedure
8
9   && ---- formulaire « client » : fin ----

```

Le bouton **Câbler le code** ajoute ce bloc s'il manque et, dans le programme principal, l'appel `FORMCLIENT_OUVRIR()` suivi de `interface.executer()` (placé avant un `ui.executer()` s'il y en a déjà un). Le programme est alors prêt à afficher le formulaire.

• Utiliser le formulaire dans le programme

Une fois `FORMCLIENT_OUVIR()` appelée, les contrôles s'utilisent par la variable globale, comme ceux que vous construisez à la main :

SYNTAXE Lire et modifier les contrôles d'un formulaire

```

1 FORMCLIENT_OUVIR()           && ouvre la fenêtre, remplit formclient
2 formclient.titre.libelle = "Modifier un client"
3 formclient.nom.valeur = "DUPONT"   && lire ou écrire la valeur d'une saisie
4 formclient.enregistrer.aktif = faux && griser un bouton
5 formclient.nom.focus()           && donner le focus
6 ? formclient.ville:element(formclient.ville:selection())
7 interface.executer()           && traite les événements jusqu'à la fermeture

```

★ À retenir

Ne modifiez jamais `form_<nom>.prg` : il est réécrit à la prochaine retouche du dessin. Tout ce qui change pendant l'exécution (valeurs, couleurs, contrôles grisés) se fait dans votre code, par `form<nom>.contrôle.propriété`.

1.9 Exemple complet: la fiche client

L'exemple `exemples/formulaire` réunit tout ce chapitre. La fenêtre a été dessinée dans le concepteur : une disposition `bordure`, un titre en haut, un cadre `identite` en `grille` au centre, une rangée de boutons en bas. Le programme principal ouvre le formulaire, puis affiche les clients saisis quand on le ferme ; les quatre procédures répondent aux événements.

EXEMPLE La fiche client principal.prg

```

1 && Exemple du concepteur de formulaires de wdlaxis+.
2 &&
3 && La fenêtre est dessinée dans le concepteur (Conception > Ouvrir un formulaire...,
4 && ou double-clic sur form_client.prg dans l'arbre du projet) ; le concepteur
5 && régénère form_client.prg à chaque modification. Ici : le programme principal
6 && et les procédures des événements, créées par « + nouvelle procédure... ».
7
8 clients = {}
9
10 FORMCLIENT_OUVIR()
11 interface.executer()
12
13 ? chaine(#clients) + " client(s) saisi(s)"
14 pour chaque c dans clients
15   ? " - ", c.prenom, c.nom, "(" + c.ville + ", " + c.categorie + ")"
16 suivant
17
18 && ---- formulaire « client » : début (généré par le concepteur, gardez ces marques) ----
19 && FORMCLIENT_OUVIR() ouvre le formulaire ; ses contrôles sont dans formclient
   (formclient.fenetre...).
20
21 procedure CLIENT_OUVERTURE(ev)

```

```

22  && surouverture de formclient (formulaire « client »)
23  formclient.nom:focus()
24  finprocedure
25
26  procedure CLIENT_SAISIE(ev)
27  && surchangement de nom (formulaire « client »)
28  formclient.enregistrer.actif = rogner(formclient.nom.valeur) <> ""
29  finprocedure
30
31  procedure CLIENT_EFFACER(ev)
32  && surclik de effacer (formulaire « client »)
33  pour chaque nom dans {"nom", "prenom", "courriel"}
34  formclient[majuscule(nom)].valeur = ""
35  suivant
36  formclient.etat.libelle = ""
37  formclient.enregistrer.actif = faux
38  formclient.nom:focus()
39  finprocedure
40
41  procedure CLIENT_ENREGISTRER(ev)
42  && surclik de enregistrer (formulaire « client »)
43  locale f = formclient
44  locale c = {nom = majuscule(rogner(f.nom.valeur)), prenom = rogner(f.prenom.valeur)}
45  c.courriel = rogner(f.courriel.valeur)
46  c.ville = f.ville:element(f.ville:selection())
47  c.categorie = sisinon(nombre(f.categorie.valeur) = 1, "particulier", "professionnel")
48  c.lettre = f.lettre:coche(1)
49  inserer(clients, c)
50  CLIENT_EFFACER(ev)
51  f.etat.libelle = c.prenom + " " + c.nom + " enregistré (" + #clients + ")"
52  finprocedure
53
54  && ---- formulaire « client » : fin ----

```

Décortiquons ce programme :

1. `FORMCLIENT_OUVRIER()` ouvre la fenêtre et `interface.executer()` attend qu'on la ferme ; la boucle `pour chaque` qui suit n'est exécutée qu'après la fermeture.
2. `CLIENT_OUVERTURE` (événement `surouverture` de la fenêtre) place le curseur dans le nom.
3. `CLIENT_SAISIE` (`surchangement` de la saisie `nom`) n'active **Enregistrer** que si le nom n'est pas vide : le bouton a été dessiné grisé.
4. `CLIENT_ENREGISTRER` lit chaque contrôle : `valeur` d'une saisie, élément choisi d'un combo (`:element(:selection())`), numéro du bouton radio choisi, case cochée (`:coche(1)`).

Voici la fiche à l'exécution, remplie puis enregistrée : la barre du bas confirme l'enregistrement, les champs sont vidés et **Enregistrer** est de nouveau grisé.

Fiche client

Nouveau client

Identité

Nom Dupont

Prénom Marie

Courriel marie.dupont@exemple.fr

Ville Brest

Catégorie

☐ Particulier

☒ Professionnel

Informations

☐ Recevoir la lettre d'information

Effacer Enregistrer

La saisie : le bouton Enregistrer s'est activé

Fiche client

Nouveau client

Identité

Nom Dupont

Prénom Marie

Courriel marie@exemple.fr

Ville Brest

Catégorie

☐ Particulier

☒ Professionnel

Informations

☐ Recevoir la lettre d'information

Marie DUPONT enregistré (1) Effacer Enregistrer

Après le clic : « Marie DUPONT enregistré (1) »



À vous de jouer

Ajoutez un champ **Téléphone** : glissez une **étiquette** et une **saisie** dans le cadre **identité**, nommez la saisie **telephone**, puis lisez **formclient.telephone.valeur** dans **CLIENT_ENREGISTRER** et ajoutez-le à l'affichage final.

1.10

Vos propres contrôles dans la palette

Vous utilisez souvent le même bouton vert, ou une étiquette mise en forme d'une façon précise ? Écrivez une **fonction** qui le crée, puis déclarez-la dans la palette : le concepteur la propose comme un contrôle à part entière.

EXEMPLE

Un bouton « Valider » prêt à l'emploi

contrôles.prg

```
1 && Contrôles personnalisés du projet (déclarés dans wdlaxis_palette.json)
2 fonction BOUTONVALIDER(libelle)
3   locale b = interface.bouton(libelle, {default = vrai})
4   b.couleur = "#FFFFFF"
5   b.fond = "#1F9D55"
6   retourner b
```

Le bouton **Contrôle personnalisé...** du concepteur demande son nom, le contrôle dont il dérive (pour l'aperçu, les propriétés et les événements) et l'expression qui le crée. Il l'enregistre dans **wdlaxis_palette.json**, à la racine du projet :

GÉNÉRÉ

La palette du projet

wdlaxis_palette.json

```
1 [
2   {"kind": "BoutonValider", "base": "bouton", "ctor": "BOUTONVALIDER($libelle)",
3     "category": "Personnalisés", "letter": "V"}
4 ]
```

Dans l'expression, `$libelle` est remplacé par la valeur de la propriété `libelle` du contrôle posé. On peut aussi ajouter des propriétés propres au contrôle (type `texte`, `entier`, `nombre`, `logique`, `couleur` ou `liste`), que l'inspecteur affichera :

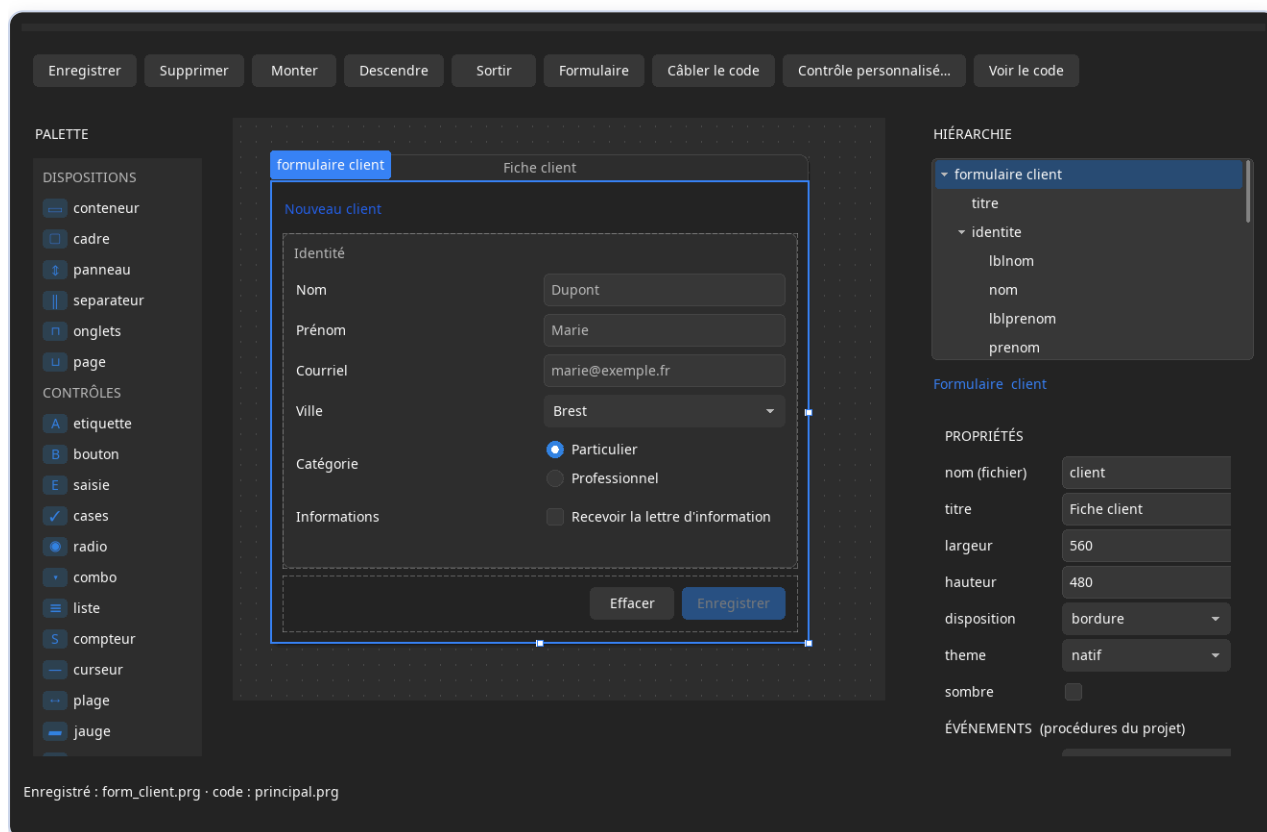
```

1 [
2   {"kind": "Pastille", "base": "etiquette", "ctor": "PASTILLE($libelle, $teinte)",
3     "props": [{"name": "teinte", "type": "couleur", "default": "#2F5BE0"}]}
4 ]

```

1.11 Le concepteur dans l'atelier en Lexis+

L'atelier écrit en Lexis+ (`exemples/atelier`, fourni avec `wdlexis+`) a le même concepteur, dans `concepteur.prg`. Le canevas et la palette sont deux briques natives du paquet `interface` : `interface.concepteur` et `interface.palette`. Le modèle et la génération de `form_<nom>.prg` sont natifs eux aussi ; la fenêtre, la hiérarchie, l'inspecteur, les procédures écrites dans le code, le câblage et le renommage sont écrits en Lexis+.



Le concepteur de l'atelier écrit en Lexis+, ouvert sur la même fiche client

Ces deux briques suffisent pour construire votre propre outil. Ce mini-concepteur rouvre la fiche client, affiche la palette, décrit le contrôle cliqué et enregistre le formulaire :



EXEMPLE Un mini-concepteur principal.prg

```

1 && Un mini-concepteur de formulaires en quelques lignes
2 ui = paquet("interface")
3 fen = ui.fenetre{titre = "Mini-concepteur", largeur = 1000, hauteur = 600, disposition =
  "bordure"}
4 c = ui.concepteur{projet = systeme.dossierprogramme(), fichier = "form_client.prg"}
5
6 pal = fen:ajouter(ui.paLETTE(c))          && la palette, liée au canevas
7 pal.ancrage = "gauche"
8 pal.largeurfixe = 180
9 bas = fen:ajouter(ui.rangee())
10 bas.ancrage = "bas"
11 infos = bas:ajouter(ui.etiquette("Cliquez sur un contrôle du formulaire"))
12 infos.poids = 1
13 benr = bas:ajouter(ui.bouton("Enregistrer"))
14 benr.surclic = @ENREGISTRER
15 zone = fen:ajouter(ui.panneau{defilement = "automatique"})
16 zone.ancrage = "centre"
17 zone:ajouter(c)          && le canevas : aperçu réel, poignées, repères
18
19 c.surselection = @CHOIX
20 ui.executer()
21
22 procedure CHOIX(ev)
23   locale e = c:element()          && l'élément sélectionné
24   infos.libelle = e.genre + " « " + e.nom + " » : " + #c:proprietes() + " propriétés, " + ;
25                 #c:evenements() + " événements"
26   retourner
27
28 procedure ENREGISTRER(ev)
29   fichier.ecrire("form_" + c:nom() + ".prg", c:code())
30   infos.libelle = "form_" + c:nom() + ".prg enregistré"
31   retourner

```

► CE QU'AFFICHE LE PROGRAMME

The screenshot shows a window titled "Fiche client" with a sub-header "Nouveau client". On the left, there is a palette titled "DISPOSITIONS" with icons for "conteneur", "cadre", "panneau", "separateur", "onglets", "page", "CONTRÔLES" with icons for "etiquette", "bouton", "saisie", "cases", "radio", "combo", "liste", and "compteur". The main form area contains the following fields:

- Identité: saisie nom (with a tooltip "saisie nom")
- Nom: Dupont
- Prénom: Marie
- Courriel: marie@exemple.fr
- Ville: Brest (dropdown menu)
- Catégorie: ☒ Particulier, ☐ Professionnel
- Informations: ☐ Recevoir la lettre d'information

At the bottom right, there are two buttons: "Effacer" and "Enregistrer". A status bar at the bottom left indicates "saisie « nom » : 12 propriétés, 10 événements".

Après un clic sur la saisie du nom, la barre du bas affiche «saisie «nom»: 12 propriétés, 10 événements». La liste complète des méthodes du concepteur (`c:proprietes()`, `c:fixerpropriete()`, `c:associer()` ...) est dans l'annexe A.



Astuce

Le canevas ne touche pas au disque: c'est votre programme qui écrit `c:code()` dans `form_<nom>.prg`, au moment qu'il choisit.

Produire des états PDF

Dessinez une facture ou un inventaire au millimètre, puis remplissez-le en quelques lignes : la pagination, les en-têtes, les pieds et les numéros de page sont calculés pour vous.



2.1 Qu'est-ce qu'un état ?

2.2 Ouvrir l'éditeur

2.3 La fenêtre de l'éditeur

2.4 Les zones

2.5 Les éléments

2.6 Les variables

2.7 Le fichier produit

2.8 Le paquet rapport

2.9 La facture

2.10 L'inventaire

2.11 Aperçu et code

2.12 Dans l'atelier en Lexis+

Dans ce chapitre, vous apprendrez à...

- ✓ découper un état en zones d'en-tête, de détail et de pied
- ✓ afficher des valeurs avec les `{variables}`
- ✓ changer d'en-tête en cours de route
- ✓ vérifier la mise en page avec l'aperçu PDF
- ✓ poser des textes, cadres, traits et images au millimètre
- ✓ remplir l'état avec `rapport.debut` et `r.ajoute`
- ✓ obtenir des zones extensibles et des sauts de page
- ✓ construire un éditeur avec `interface.etat`

2.1 Qu'est-ce qu'un état ?

Un **état** est un document imprimable produit par le programme : une facture, une liste de clients, un inventaire, un bon de livraison. Sa particularité : on ne sait pas d'avance combien de lignes il aura, donc combien de pages. On ne le dessine donc pas page par page, mais **zone par zone**.



Les zones, comme des tampons

Imaginez un jeu de tampons encreurs : un tampon « en-tête » avec le logo, un tampon « ligne » avec une case pour la désignation et une pour le prix, un tampon « total ». Le programme tamponne la ligne autant de fois qu'il y a d'articles, les unes sous les autres ; quand la page est pleine, il prend une nouvelle feuille, y tamponne l'en-tête et le pied, et continue.



Zone	Rôle
entete	imprimée en haut de chaque page générée
pied	imprimée en bas de chaque page générée
tout autre nom : {titre}, {ligne}, {total} ...	ajoutée par le programme, autant de fois qu'il le faut, les unes sous les autres ; une zone qui ne tient plus dans la page passe à la suivante, qui reçoit à son tour l'en-tête et le pied

Chaque zone contient des **éléments** placés au millimètre : des textes, des rectangles, des lignes et des images. Les textes peuvent contenir des **variables** entre accolades, comme **{client}**, que le programme remplit.

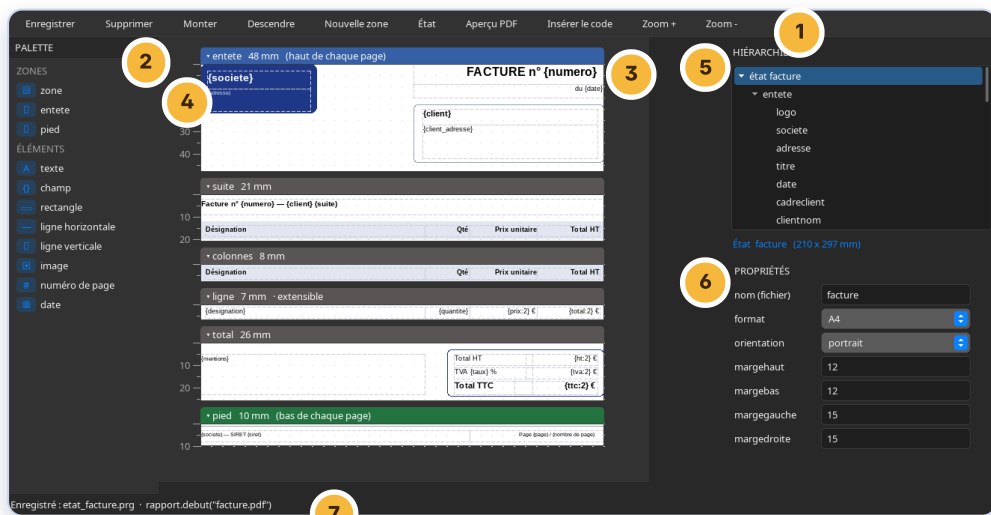
2.2 Ouvrir l'éditeur d'états

Commande	Effet
Conception > Nouvel état...	crée un état neuf, nommé etat1 , etat2 ... (le premier libre)
Conception > Ouvrir un état...	rouvre un etat_<nom>.prg existant
double-clic sur etat_<nom>.prg	dans l'arbre du projet : même effet

Un état neuf n'est pas vide : il contient un en-tête (un titre **{titre}**, la date « Le {date} » et un trait), une zone **ligne** (**{libelle}** à gauche, **{valeur}** à droite) et un pied qui numérote les pages (« Page {page} / {nombre de page} »). Il s'imprime donc déjà : il ne reste qu'à l'adapter.

2.3 La fenêtre de l'éditeur

Voici l'éditeur ouvert sur l'état **facture** de l'exemple **exemples/etats**. Les zones sont dessinées à l'échelle, à la largeur de la page (marges déduites), avec une règle en millimètres à gauche.



- | | |
|--|-------------------------------|
| 1 Barre d'outils : zones, aperçu, code, zoom | 2 Palette : zones et éléments |
| 3 Les zones, avec leur nom et leur hauteur | 4 Règle en millimètres |
| 5 Hiérarchie : l'état, ses zones, leurs éléments | 6 Propriétés de la sélection |
| 7 Barre d'état : le fichier et l'appel rapport.debut | |

Bouton	Effet
Enregistrer	régénère <code>etat_<nom>.prg</code> (fait aussi à chaque modification)
Supprimer	supprime l'élément ou la zone sélectionnés (Suppr)
Monter / Descendre	une zone : plus haut ou plus bas dans l'état ; un élément : dessiné plus tôt ou plus tard
Nouvelle zone	ajoute une zone nommée après la zone sélectionnée
État	sélectionne l'état : format, orientation, marges
Aperçu PDF	génère le PDF avec chaque zone trois fois, variables affichées (2.11)
Insérer le code	écrit dans l'éditeur le code Lexis+ qui remplit l'état (2.11)
Zoom + / Zoom -	agrandit ou réduit la page

2.4 Les zones

La palette propose trois sortes de zones : **zone** (une zone nommée, à remplir par le programme), **entete** et **pied**. Glissez-les sur l'état ou cliquez-les. Le **bord bas** d'une zone se tire à la souris pour régler sa hauteur. Sélectionnée (clic sur son bandeau), une zone montre ses propriétés :

Propriété	Rôle
nom	le nom utilisé par le programme : <code>r.ajoute("ligne", ...)</code>
hauteur	en millimètres
extensible	ses textes l'agrandissent quand ils passent à la ligne, et les éléments placés dessous descendent d'autant
sautavant	la zone commence toujours une nouvelle page
fond	couleur de fond de toute la zone

L'état lui-même (bouton **État**) a un **format** (**A4**, **A5**, **A3**, **Lettre**, **Legal**), une **orientation** (**portrait** ou **paysage**) et quatre marges en millimètres (**margehaut** et **margebas** : 12 ; **margegauche** et **margedroite** : 15 par défaut).

À retenir

Une zone ne se coupe jamais entre deux pages : si elle ne tient plus, elle passe entière sur la page suivante. Gardez donc des zones de détail **basses** (une ligne d'article), et regroupez dans une zone ce qui doit rester ensemble (les totaux).

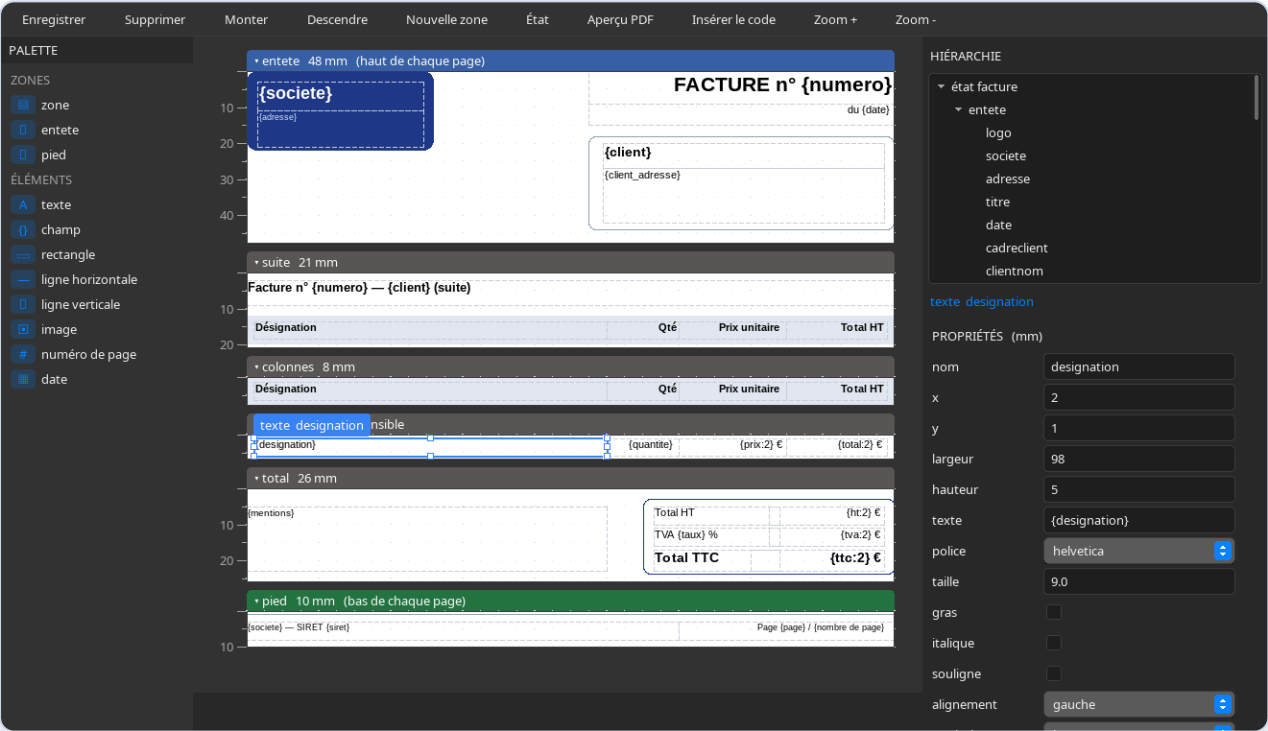
2.5 Les éléments

La palette propose huit éléments ; certains sont des textes déjà réglés.



Dans la palette	Ce qui est posé
texte	un texte fixe, « Texte »
champ	un texte contenant une variable : {champ}
rectangle	un cadre (bordure, fond, coins arrondis)
ligne horizontale / verticale	un trait, plein ou pointillé
image	une image PNG ou JPEG
numéro de page	un texte aligné à droite : « Page {page} / {nombre de page} »
date	un texte {date}

Glissez un élément sur une zone, ou cliquez-le pour l'ajouter à la zone sélectionnée. Glisser un élément le déplace, y compris d'une zone à l'autre ; ses poignées le redimensionnent. Des **repères magenta** alignent ses bords et son centre sur ceux des autres éléments et de la zone ; sinon les positions vont de 0,5 mm en 0,5 mm. Au clavier : **Suppr** supprime, les **flèches** déplacent de 0,5 mm (**Maj** : 5 mm), **Échap** sélectionne la zone, puis l'état.



Le texte **designation** de la zone **ligne** sélectionné (zoom 125 %) : sa position, sa taille et sa mise en forme

● **Les propriétés d'un texte**

Propriété	Rôle (valeur par défaut)
x, y, largeur, hauteur	position et taille dans la zone, en millimètres
texte	le texte, avec ses {variables}
police	helvetica, times ou courier (helvetica)



Propriété	Rôle (valeur par défaut)
taille	en points (10)
gras, italique, souligne	style (non)
alignement	<code>gauche</code> , <code>centre</code> , <code>droite</code> (gauche)
vertical	<code>haut</code> , <code>milieu</code> , <code>bas</code> (haut)
retour	retour à la ligne automatique (oui)
couleur, fond, bordure	couleurs du texte, du fond et du cadre (noir, aucun, aucun)
epaisseur	épaisseur du cadre, en points (0,5)

• Rectangles, lignes et images

Élément	Propriétés (valeur par défaut)
rectangle	<code>bordure</code> (noir), <code>epaisseur</code> (0,5), <code>fond</code> (aucun), <code>rayon</code> des coins en mm (0)
ligne	<code>couleur</code> (noir), <code>epaisseur</code> (0,5), <code>pointille</code> (non)
image	<code>chemin</code> (relatif au dossier du programme ; peut contenir une variable), <code>mode</code> : <code>ajuster</code> (proportions gardées), <code>etirer</code> ou <code>centre</code>



Astuce

Le texte est mis en page dans l'éditeur avec les mêmes mesures que dans le PDF : ce que vous voyez à l'écran, coupures de ligne comprises, est ce qui sera imprimé. Les polices standard des PDF (Helvetica, Times, Courier) couvrent tous les caractères du français : é, ç, œ, « », €...

2.6 Les variables

Dans un texte (et dans le chemin d'une image), un nom entre accolades est remplacé par la valeur que le programme donne pour cette zone. Quatre variables sont calculées par le générateur lui-même.

SYNTAXE Les variables d'un état	
<code>{client}</code>	valeur donnée par le programme (majuscules indifférentes)
<code>{prix:2}</code>	nombre avec 2 décimales : 1 234,50
<code>{page}</code>	numéro de la page
<code>{nombre de page}</code>	nombre total de pages (aussi <code>{nombre de pages}</code> , <code>{pages}</code>)
<code>{date}</code>	date du jour : 25/10/2026
<code>{heure}</code>	heure : 14:05

- Les noms ne tiennent pas compte des majuscules : `{Client}` et `{client}` reçoivent la même valeur.
- Une variable que le programme ne donne pas est remplacée par un texte vide.
- `:2` (ou `:0`, `:3` ...) formate un nombre à la française : virgule décimale, espace entre les milliers. Sans format, un nombre s'affiche tel quel (`4.9`) : utilisez toujours un format pour les montants.

★ À retenir

`{page}` et `{nombre de page}` ne sont connus qu'à la fin : c'est `r.fin()` qui les calcule, une fois toutes les zones placées. « Page 2 / 5 » s'écrit simplement `Page {page} / {nombre de page}`.

2.7 Le fichier produit par l'éditeur

L'état est enregistré à chaque modification dans `etat_<nom>.prg` : une fonction `ETAT_<NOM>()` qui renvoie sa description. Il est donc chargé avec les autres fichiers du projet, et embarqué dans les applications générées. Son en-tête rappelle les zones à remplir.

```
● ● ● GÉNÉRÉ Le début de l'état « facture » etat_facture.prg

1 && État « facture » – généré par l'éditeur d'états de wdllexis+.
2 && Régénéré à chaque modification dans l'éditeur : modifiez le dessin là-bas, pas ce fichier.
3 &&   r = rapport.debut("facture.pdf")
4 &&   r.ajoute("zone", {variable = valeur, ...})      zones de l'état : entete, suite,
   colonnes, ligne, total, pied
5 &&   r.fin()                                          écrit le PDF
6
7 fonction ETAT_FACTURE()
8 retourner [[
9 {
10  "etat": "facture",
11  "props": {},
12  "zones": [
13    {
14      "name": "entete",
15      "height": 48.0,
16      "props": {},
17      "items": [
18        {
19          "kind": "rectangle",
20          "name": "logo",
21          "x": 0.0,
22          "y": 0.0,
23          "w": 52.0,
24          "h": 22.0,
25          "props": {
26            "fond": "#1E3A8A",
27            "bordure": "",
28            "rayon": "3"
29          }
30        },
31        {
32          ...
```

2.8 Remplir l'état : le paquet rapport

Le paquet `rapport` est disponible par son nom, sans `paquet()`. Écrit entièrement en Nim, sans bibliothèque extérieure, il fonctionne dans toutes les applications générées, avec ou sans interface graphique : console, service ou bibliothèque.

SYNTAXE Le paquet rapport

```

1 r = rapport.debut(pdf, [etat])      && ou rapport.début(...) : commence un état
2 r.ajoute(zone, {variable = valeur, ...}) && ajoute une zone à la suite
3 r.entete(zone, {variable = valeur, ...}) && ou r.entête(...) : remplace l'en-tête
4 r.pied(zone, {variable = valeur, ...}) && remplace le pied
5 chemin = r.fin()                   && calcule {page}, {nombre de page}, écrit le PDF
6 n = r.nombrepages()                && nombre de pages (définitif après r.fin())
7 liste = rapport.etats()             && noms des états du projet

```

- `rapport.debut(pdf, [etat])` commence un état et renvoie l'état en cours, `r`. Sans `etat`, l'état porte le nom du PDF sans `.pdf` : `rapport.debut("facture.pdf")` utilise `etat_facture.prg`. Un chemin relatif est pris dans le dossier du programme.
- `r.ajoute(zone, {...})` place la zone sous la précédente, avec ses variables ; si elle ne tient plus, elle ouvre une nouvelle page, avec l'en-tête et le pied.
- Au départ, l'en-tête est la zone `entete` et le pied la zone `pied`, sans variables. `r.entete` et `r.pied` donnent leurs variables, ou les remplacent par une autre zone.
- `r.fin()` calcule les numéros de page, écrit le PDF et renvoie son chemin.

● Changer d'en-tête en cours de route

Une facture a souvent un grand en-tête sur la première page, et un rappel plus discret sur les suivantes.

`r.entete(zone, {...})` remplace l'en-tête **pour la page en cours si rien n'y est encore placé**, sinon à **partir de la page suivante**. `r.pied` remplace le pied pour la page en cours et les suivantes.

SYNTAXE Un en-tête pour la première page, un autre pour la suite

```

1 r = rapport.debut("facture.pdf")
2 r.entete("entete", {numero = "2026-0142"}) && 1re page : l'en-tête « entete »
3 r.ajoute("colonnes", {}) && la page 1 a déjà reçu une zone...
4 r.entete("suite", {numero = "2026-0142"}) && ... donc « suite » vaut à partir de la page 2
5 r.entete(nul) && plus d'en-tête du tout

```



Astuce

Lexis+ accepte les deux orthographes : `rapport.début` et `rapport.debut`, `r.entête` et `r.entete`. Il accepte aussi `r.ajoute(...)` comme `r.ajoute(...)`, et `{{variable = valeur}}` comme `{variable = valeur}`.

2.9 Exemple complet: une facture de deux pages

L'exemple `exemples/etats` produit une facture de 42 articles. L'état `facture` (figure de la section 2.3) a six zones : `entete` (le cadre de la société, le numéro, la date, le client), `suite` (l'en-tête des pages suivantes, avec les titres des colonnes), `colonnes`, `ligne` (extensible: une longue désignation l'agrandit), `total` et `pied`. Le programme ne fait que donner les valeurs.

```

1 && Exemple d'état PDF : une facture de plusieurs pages.
2 &&
3 && La mise en page est dessinée dans l'éditeur d'états (Conception > Ouvrir un
4 && état..., ou double-clic sur etat_facture.prg) : zones « entete » (première
5 && page), « suite » (en-tête des pages suivantes), « colonnes », « ligne »
6 && (extensible : une longue désignation l'agrandit), « total » et « pied ».
7 && Ici, le programme ne fait que donner les valeurs des {variables}.
8
9 societe = {societe = "Atelier Kerlann", adresse = "12 quai de la Douane" + car(10) + "29200
   Brest", ;
10         siret = "123 456 789 00012"}
11 taux = 20
12
13 articles = {}
14 pour i = 1 a 42
15     inserer(articles, {designation = "Article de démonstration n° " + i, quantite = 1 + i % 4,
   prix = 9.5 + i * 1.25})
16 suivant
17 articles[7].designation = "Prestation de mise en service sur site, avec déplacement, réglages,
   essais " + ;
18         "et formation de deux personnes à l'utilisation (demi-journée)"
19
20 r = rapport.début("facture.pdf")
21 r.entete("entete", {societe = societe.societe, adresse = societe.adresse, numero = "2026-0142",
   ;
22         client = "Librairie du Port", client_adresse = "3 rue de Siam" + car(10) +
   "29200 Brest"})
23 r.pied("pied", {societe = societe.societe, siret = societe.siret})
24 r.ajoute("colonnes", {})
25 && à partir de la page 2 : l'en-tête « suite »
26 r.entete("suite", {numero = "2026-0142", client = "Librairie du Port"})
27
28 ht = 0
29 pour chaque a dans articles
30     locale total = a.quantite * a.prix
31     ht += total
32     r.ajoute("ligne", {designation = a.designation, quantite = a.quantite, prix = a.prix, total =
   total})
33 suivant
34
35 r.ajoute("total", {ht = ht, taux = taux, tva = ht * taux / 100, ttc = ht * (1 + taux / 100), ;
36         mentions = "Paieement à 30 jours. Pas d'escompte pour paiement anticipé."})
37 chemin = r.fin()
38 ? "Facture écrite :", chemin, "(" + r.nombrepages() + " pages)"

```

► CE QU'AFFICHE LE PROGRAMME

Facture écrite : .../etats/facture.pdf (2 pages)

Et voici les deux pages du PDF produit :

Atelier Kerlann

12 quai de la Douane
29200 Brest

FACTURE n° 2026-0142

du 25/10/2026

Librairie du Port

3 rue de Siam
29200 Brest

Désignation	Qté	Prix unitaire	Total HT
Article de démonstration n° 1	2	10,75 €	21,50 €
Article de démonstration n° 2	3	12,00 €	36,00 €
Article de démonstration n° 3	4	13,25 €	53,00 €
Article de démonstration n° 4	1	14,50 €	14,50 €
Article de démonstration n° 5	2	15,75 €	31,50 €
Article de démonstration n° 6	3	17,00 €	51,00 €
Prestation de mise en service sur site, avec déplacement, réglages, essais et formation de deux personnes à l'utilisation (demi-journée)	4	18,25 €	73,00 €
Article de démonstration n° 8	1	19,50 €	19,50 €
Article de démonstration n° 9	2	20,75 €	41,50 €
Article de démonstration n° 10	3	22,00 €	66,00 €
Article de démonstration n° 11	4	23,25 €	93,00 €
Article de démonstration n° 12	1	24,50 €	24,50 €
Article de démonstration n° 13	2	25,75 €	51,50 €
Article de démonstration n° 14	3	27,00 €	81,00 €
Article de démonstration n° 15	4	28,25 €	113,00 €
Article de démonstration n° 16	1	29,50 €	29,50 €
Article de démonstration n° 17	2	30,75 €	61,50 €
Article de démonstration n° 18	3	32,00 €	96,00 €
Article de démonstration n° 19	4	33,25 €	133,00 €
Article de démonstration n° 20	1	34,50 €	34,50 €
Article de démonstration n° 21	2	35,75 €	71,50 €
Article de démonstration n° 22	3	37,00 €	111,00 €
Article de démonstration n° 23	4	38,25 €	153,00 €
Article de démonstration n° 24	1	39,50 €	39,50 €
Article de démonstration n° 25	2	40,75 €	81,50 €
Article de démonstration n° 26	3	42,00 €	126,00 €
Article de démonstration n° 27	4	43,25 €	173,00 €
Article de démonstration n° 28	1	44,50 €	44,50 €
Article de démonstration n° 29	2	45,75 €	91,50 €

Paiement à 30 jours. Pas d'escompte pour paiement anticipé.

Total HT

TVA 20 %

Total TTC

3 807,50 €

761,50 €

4 569,00 €

Atelier Kerlann — SIRET 123 456 789 00012

Page 1 / 2

Facture n° 2026-0142 — Librairie du Port (suite)

Désignation	Qté	Prix unitaire	Total HT
Article de démonstration n° 30	3	47,00 €	141,00 €
Article de démonstration n° 31	4	48,25 €	193,00 €
Article de démonstration n° 32	1	49,50 €	49,50 €
Article de démonstration n° 33	2	50,75 €	101,50 €
Article de démonstration n° 34	3	52,00 €	156,00 €
Article de démonstration n° 35	4	53,25 €	213,00 €
Article de démonstration n° 36	1	54,50 €	54,50 €
Article de démonstration n° 37	2	55,75 €	111,50 €
Article de démonstration n° 38	3	57,00 €	171,00 €
Article de démonstration n° 39	4	58,25 €	233,00 €
Article de démonstration n° 40	1	59,50 €	59,50 €
Article de démonstration n° 41	2	60,75 €	121,50 €
Article de démonstration n° 42	3	62,00 €	186,00 €

Paiement à 30 jours. Pas d'escompte pour paiement anticipé.

Total HT

TVA 20 %

Total TTC

3 807,50 €

761,50 €

4 569,00 €

Atelier Kerlann — SIRET 123 456 789 00012

Page 2 / 2

Observez :

- la page 1 porte l'en-tête `entete` ; la page 2 porte `suite` , car `r.entete("suite", ...)` a été appelé après la première zone ;
- l'article 7, à la désignation très longue, occupe deux lignes : la zone `ligne` est **extensible**, et son trait de séparation est descendu d'autant ;
- les montants sont formatés par `{prix:2}` (« 1 234,50 ») ;
- le pied affiche « Page 1 / 2 » et « Page 2 / 2 » : le total des pages a été calculé par `r.fin()` .

2.10

Exemple complet: un inventaire par rayon

Deuxième exemple, une liste regroupée : l'inventaire d'une papeterie, rayon par rayon. L'état `stock` a une zone `rayon` (le titre du groupe), une zone `article` (extensible), une zone `soustotal` , et une zone `synthese` dont la propriété **sautavant** est cochée : elle commence toujours une nouvelle page.



EXEMPLE

L'inventaire du stock

principal.prg

```
1 && Inventaire du stock, rayon par rayon, dans un état PDF
2 rayons = {
3   {nom = "Papeterie", articles = {
4     {reference = "PAP-001", article = "Ramette papier A4 80 g (500 feuilles)", quantite = 140,
      prix = 4.9},
5     {reference = "PAP-014", article = "Cahier 96 pages grands carreaux", quantite = 320, prix =
      1.65},
```

```

6      {reference = "PAP-027", article = "Bloc-notes repositionnables 76 x 76", quantite = 85,
prix = 2.3}}},
7      {nom = "Écriture", articles = {
8          {reference = "ECR-003", article = "Stylo bille bleu pointe moyenne", quantite = 600, prix =
0.45},
9          {reference = "ECR-019", article = "Surligneur jaune fluo", quantite = 210, prix = 0.95},
10         {reference = "ECR-022", article = "Porte-mine 0,5 mm avec gomme", quantite = 48, prix =
3.2}}},
11     {nom = "Classement", articles = {
12         {reference = "CLA-008", article = "Classeur à levier dos 80 mm, carton rigide recouvert de
polypropylène, coloris assortis", quantite = 75, prix = 3.9},
13         {reference = "CLA-011", article = "Pochettes perforées (100)", quantite = 120, prix =
5.4}}}}
14
15 r = rapport.début("stock.pdf")
16 r.entête("entete", {magasin = "Papeterie du Port – Brest"})
17 r.pied("pied", {magasin = "Papeterie du Port"})
18 r.ajoute("colonnes", {})
19
20 total = 0
21 nb = 0
22 pour chaque ry dans rayons
23     r.ajoute("rayon", {rayon = ry.nom})
24     locale montant = 0
25     pour chaque a dans ry.articles
26         locale valeur = a.quantite * a.prix
27         montant += valeur
28         nb += 1
29         r.ajoute("article", {reference = a.reference, article = a.article, quantite = a.quantite, ;
30             prix = a.prix, valeur = valeur})
31     suivant
32     r.ajoute("soustotal", {rayon = ry.nom, montant = montant})
33     total += montant
34 suivant
35
36 r.ajoute("synthese", {rayons = #rayons, articles = nb, total = total})    && sautavant :
nouvelle page
37 ? "État écrit :", r.fin()
38 ? "Pages :", r.nombrepages()
39 ? "Valeur du stock :", total

```

► CE QU'AFFICHE LE PROGRAMME

État écrit : .../stock/stock.pdf
 Pages : 2
 Valeur du stock : 2973.1



Inventaire du stock				
Papeterie du Port — Brest				
Arrêté au 25/10/2026				
Référence	Article	Qté	Prix	Valeur
Papeterie				
PAP-001	Ramette papier A4 80 g (500 feuilles)	140	4,90 €	686,00 €
PAP-014	Cahier 96 pages grands carreaux	320	1,65 €	528,00 €
PAP-027	Bloc-notes repositionnables 76 x 76	85	2,30 €	195,50 €
		Total Papeterie		1 409,50 €
Ecriture				
ECR-003	Stylo bille bleu pointe moyenne	600	0,45 €	270,00 €
ECR-019	Surigneur jaune fluo	210	0,95 €	199,50 €
ECR-022	Porte-mine 0,5 mm avec gomme	48	3,20 €	153,60 €
		Total Ecriture		623,10 €
Classement				
CLA-008	Classeur à levier dos 80 mm, carton rigide recouvert de polypropylène, coloris assortis	75	3,90 €	292,50 €
CLA-011	Pochettes perforées (100)	120	5,40 €	648,00 €
		Total Classement		940,50 €

Inventaire du stock	
Papeterie du Port — Brest	
Arrêté au 25/10/2026	
Synthèse	
Rayons	3
Références en stock	8
Valeur totale	2 973,10 €

Le programme calcule lui-même les sous-totaux et le total : l'état ne fait qu'afficher ce qu'on lui donne. C'est la bonne répartition des rôles : les calculs dans le code, la présentation dans l'état.



À vous de jouer

Ajoutez une zone `alerte`, au fond orangé, avec le texte « Stock bas : {article} ». Dans la boucle, après chaque article, ajoutez-la avec `r.ajoute("alerte", {article = a.article})` quand la quantité est inférieure à 50.

2.11

Vérifier : l'aperçu PDF et le code inséré

Deux boutons de l'éditeur aident à mettre au point un état sans écrire de programme :

- **Aperçu PDF** génère l'état avec **chaque zone trois fois**, les variables affichées telles quelles (`{client}`), et l'ouvre dans la visionneuse PDF de l'atelier. On y vérifie l'enchaînement des zones, les marges et la place des éléments.
- **Insérer le code** écrit dans l'éditeur, au curseur, le code qui remplit l'état : `rapport.debut`, un `r.ajoute` par zone avec toutes ses variables, et `r.fin()`. Il ne reste qu'à remplacer les valeurs.

{societe}

(adresse)

FACTURE n° {numero}

du 25/10/2026

{client}

{client_adresse}

Facture n° {numero} — {client} (suite)

Désignation	Qté	Prix unitaire	Total HT
Facture n° {numero} — {client} (suite)			
Désignation	Qté	Prix unitaire	Total HT
Facture n° {numero} — {client} (suite)			
Désignation	Qté	Prix unitaire	Total HT
Facture n° {numero} — {client} (suite)			
Désignation	Qté	Prix unitaire	Total HT
{designation}	{quantite}	{prix} €	{total} €
{designation}	{quantite}	{prix} €	{total} €
{designation}	{quantite}	{prix} €	{total} €

(mentions)

Total HT

{ht} €

TVA (taux) %

{tva} €

Total TTC

{ttc} €

(mentions)

Total HT

{ht} €

TVA (taux) %

{tva} €

Total TTC

{ttc} €

(mentions)

Total HT

{ht} €

TVA (taux) %

{tva} €

Total TTC

{ttc} €

{societe} — SIRET {siret}

Page 1 / 1

L'aperçu de l'état « facture » : chaque zone trois fois, variables visibles



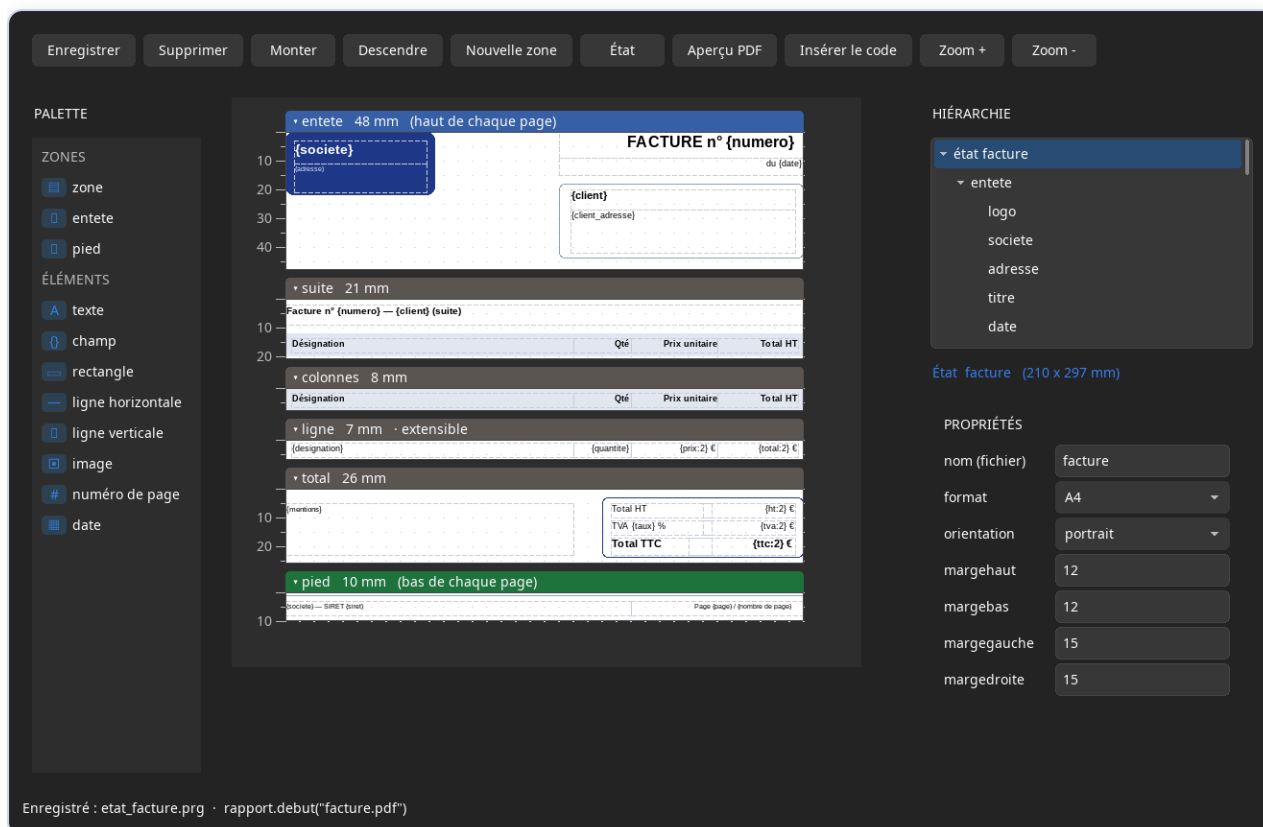
Attention

Dans l'aperçu, une variable s'affiche `{nom}` ; dans le vrai PDF, une variable que le programme oublie de donner devient un texte vide. Si un champ manque à l'impression, vérifiez l'orthographe de la variable dans `r.ajoute`.

2.12

L'éditeur d'états dans l'atelier en Lexis+

L'atelier écrit en Lexis+ a le même éditeur, dans `etats.prg`. Le canevas `interface.etat`, la palette `interface.paletteetat`, le modèle, la mise en page et le PDF sont natifs ; la fenêtre, la hiérarchie, l'inspecteur, le renommage, l'aperçu et l'insertion du code sont écrits en Lexis+. Les fichiers `etat_<nom>.prg` sont les mêmes.



L'éditeur d'états de l'atelier écrit en Lexis+

Comme pour les formulaires, ces briques suffisent à construire un outil. Ce mini-éditeur rouvre la facture, décrit l'élément cliqué et produit un aperçu :



```

23  infos.libelle = s.genre + " « " + s.nom + " » : " + #e:proprietes() + " propriétés"
24  retourner
25
26  procedure APERCU(ev)
27    locale r = e:apercu("apercu.pdf", systeme.dossierprogramme())
28    infos.libelle = "Aperçu écrit : " + fichier.nom(r.chemin) + " (" + #r.erreurs + " erreur(s))"
29  retourner

```

► CE QU'AFFICHE LE PROGRAMME

ZONES

- zone
- entete
- pied

ÉLÉMENTS

- texte
- champ
- rectangle
- ligne horizontale
- ligne verticale
- image
- # numéro de page
- date

entete 48 mm (haut de chaque page)

FACTURE n° {numero}

du {date}

{societe}

{client}

{client_adresse}

suite 21 mm

Facture n° {numero} — {client} {suite}

Désignation	Qté	Prix unitaire	Total HT
colonnes 8 mm			
Désignation	Qté	Prix unitaire	Total HT
texte designation	ensable		
{designation}	{quantite}	{prix:2} €	{total:2} €
total 26 mm			
{montants}			
Total HT		{ht:2} €	
TVA {taux} %		{tva:2} €	
Total TTC		{ttc:2} €	

pied 10 mm (bas de chaque page)

texte « designation » : 17 propriétés

Aperçu PDF

Un clic sur **Aperçu PDF** écrit **apercu.pdf** dans le dossier du programme ; la barre du bas affiche alors « Aperçu écrit : apercu.pdf (0 erreur(s)) ».

Référence **complète**

La palette, les propriétés et les événements des formulaires ; les zones, les éléments et les variables des états ; le paquet **rapport** et les briques du paquet **interface** .





Référence · Formulaires

La palette, les propriétés, les événements et la brique interface.concepteur.

■ Palette : dispositions

Contrôle	Propriétés propres (défaut)
conteneur	disposition (vertical), marge, espacement, colonnes (2)
cadre	titre (Groupe), disposition, marge, espacement, colonnes
panneau	disposition, marge, espacement, colonnes, defilement (toujours, automatique, masque)
separateur	vertical (oui), position (0,5) — événement surchangement
onglets	— événement surchangement ; contient des pages
page	titre (Page), disposition, marge, espacement, colonnes

■ Palette : contrôles

Contrôle	Propriétés propres (défaut)
etiquette	libelle (Libellé), alignement (gauche, centre, droite)
bouton	libelle (Bouton), défaut, annuler
saisie	texte, indication, multiligne
cases	options (liste)
radio	options (Un;Deux;Trois), selection (1)
combo	elements (liste), selection (1)
liste	elements (liste), multi
compteur, curseur, ascenseur	valeur, min (0), max (100)
plage	basse (20), haute (80), min, max
jauge	valeur (40), min, max
notation	valeur (3)
image	chemin
codebarre	texte (WDLEXIS)

■ Palette : données

Contrôle	Propriétés propres (défaut)
calendrier	date
graphe	genre (colonnes, lignes, aires, secteurs), titre
arbre	noeuds (Racine;Racine/Enfant)
tableau	colonnes (Nom;Valeur)
grille	colonnes (nom:Nom;valeur:Valeur), multi, modifiable
kanban	listes (À faire;En cours;Fait)
mosaïque, gantt	—
markdown	texte
barremenu	menus (Fichier/Ouvrir;Fichier/Quitter;Aide/À propos)

■ Fenêtre et propriétés communes

Propriété	Rôle
titre, largeur, hauteur	fenêtre : titre (Formulaire), 640 × 480
disposition	vertical, horizontal, grille, flux, bordure, pile, absolue
theme, sombre	fenêtre : natif, windows, macos, linux ; thème sombre
infobulle, visible	tous : texte au survol ; affiché ou non
etat	tous : actif, grise, lecture
largeurfixe, hauteurfixe	tous : taille imposée (0 : naturelle)
x, y	tous : position en disposition absolue
poids, etirer	tous : part de la place libre ; étirement
ancrage	tous : haut, bas, gauche, droite, centre (disposition bordure)
couleur, fond	tous : couleurs #RRGGBB

■ Fichiers et noms

Élément	Rôle
form_<nom>.prg	fichier du formulaire, à la racine du projet ; réécrit à chaque modification
FORM<NOM>_OUVRIR()	ouvre la fenêtre, renvoie la table de ses contrôles
form<nom>	variable globale : la même table (form<nom>.fenetre, form<nom>.<contrôle>)
&&\$...	dessin du formulaire, relu par le concepteur

Élément	Rôle
&& ---- formulaire « nom » : début / fin	marques du bloc des procédures dans le code associé
<CONTRÔLE>_<ÉVÉNEMENT>(ev)	nom proposé par « + nouvelle procédure... » (ENREGISTRER_CLIC)
wdlexis_palette.json	contrôles personnalisés : kind, base, ctor (\$propriété), category, letter, props

■ interface.concepteur et interface.palette

Appel	Rôle
interface.concepteur{projet=, fichier= nom=}	Canevas du concepteur : aperçu réel, glisser-déposer, repères, poignées. fichier : form_<nom>.prg à rouvrir ; nom : nouveau formulaire. surselection : élément sélectionné ; surchangement : dessin modifié.
interface.palette(concepteur)	Palette : glisser un contrôle sur le canevas, ou cliquer pour l'ajouter au conteneur sélectionné.
c:nom()	Nom du formulaire.
c:code()	Texte de form_<nom>.prg (fonction FORM<NOM>_OUVRIR et dessin).
c:codeassocie([chemin])	Fichier de code du formulaire (relatif au projet) ; avec un argument : le fixe.
c:noeuds()	Éléments : liste de {uid, nom, genre, chemin, niveau, conteneur}.
c:selection()	uid de l'élément sélectionné.
c:sélectionner([uid])	Sélectionne un élément (sans argument : la fenêtre).
c:element([uid])	{uid, nom, genre, racine, conteneur} d'un élément (sélectionné par défaut).
c:proprietes([uid])	Liste de {nom, genre, valeur, choix} ; genre : texte, entier, nombre, logique, choix, couleur, liste.
c:fixerpropriete(nom, valeur, [uid])	Modifie une propriété (« nom » renommé) ; nul si c'est fait, sinon le message d'erreur.
c:evenements([uid])	Événements de l'élément : liste de {nom, procedure}.
c:associer(evenement, procedure, [uid])	Associe une procédure à un événement ("" : retire l'association).
c:supprimer()	Supprime l'élément sélectionné.
c:monter() · c:descendre()	Plus tôt / plus tard dans l'ordre d'affichage.
c:sortir()	Sort l'élément de son conteneur.
c:ajouterpersonnalise(projet, genre, base, expression)	Ajoute à la palette un contrôle créé par une fonction Lexis+ (wdlexis_palette.json) ; nul ou le message d'erreur.



Référence · États PDF

Le paquet rapport, les zones, les éléments, les variables et la brique interface.etat.

■ Paquet rapport

Appel	Rôle
<code>rapport.debut(pdf, [etat])</code>	Commence un état (<code>etat_<etat>.prg</code> ; sans <code>etat</code> : le nom du PDF sans « .pdf ») ; chemin relatif : dossier du programme. « début » s'écrit aussi. Renvoie l'état en cours <code>r</code> .
<code>r.ajoute(zone, {variable = valeur, ...})</code>	Ajoute une zone à la suite ; nouvelle page (avec en-tête et pied) si elle ne tient plus.
<code>r.entete(zone, {variable = valeur, ...})</code>	Remplace l'en-tête (au départ : la zone « entete ») : pour la page en cours si rien n'y est placé, sinon à partir de la suivante ; « entête » s'écrit aussi ; zone nul : plus d'en-tête.
<code>r.pied(zone, {variable = valeur, ...})</code>	Remplace le pied (au départ : la zone « pied »), pour la page en cours et les suivantes ; zone nul : plus de pied.
<code>r.fin()</code>	Calcule {page} et {nombre de page}, écrit le PDF, renvoie son chemin.
<code>r.nombrepages()</code>	Nombre de pages (définitif après <code>r.fin()</code>).
<code>rapport.etats()</code>	Noms des états du projet.

■ L'état, les zones

Propriété	Rôle (défaut)
<code>format</code>	état : A4, A5, A3, Lettre, Legal (A4)
<code>orientation</code>	état : portrait, paysage (portrait)
<code>margehaut, margebas</code>	état : marges en mm (12)
<code>margegauche, margedroite</code>	état : marges en mm (15)
<code>entete, pied</code>	zones imprimées en haut et en bas de chaque page
<code>hauteur</code>	zone : en mm
<code>extensible</code>	zone : grandit avec ses textes ; les éléments dessous descendent (non)
<code>sautavant</code>	zone : toujours sur une nouvelle page (non)
<code>fond</code>	zone : couleur de fond (aucun)



■ Les éléments

Propriété	Rôle (défaut)
x, y, largeur, hauteur	tous : position et taille dans la zone, en mm
texte	texte : avec {variables} (Texte)
police	texte : helvetica, times, courier (helvetica)
taille	texte : en points (10)
gras, italique, souligne	texte : style (non)
alignement, vertical	texte : gauche, centre, droite ; haut, milieu, bas
retour	texte : retour à la ligne automatique (oui)
couleur, fond, bordure	texte : couleurs (#000000, aucun, aucun)
epaisseur	texte, rectangle, ligne : en points (0,5)
bordure, fond, rayon	rectangle : cadre (#000000), fond, coins en mm (0)
couleur, pointille	ligne : (#000000), pointillé (non)
chemin, mode	image : PNG ou JPEG ; ajuster, étirer, centre (ajuster)

■ Les variables

Variable	Valeur
{nom}	valeur donnée par le programme pour la zone ; vide si absente (aperçu : {nom})
{nom:2}	nombre à 2 décimales, virgule décimale, espaces des milliers
{page}	numéro de la page
{nombre de page}	nombre de pages ; aussi {nombre de pages}, {pages}
{date} · {heure}	jj/mm/aaaa · hh:mm

■ interface.etat et interface.paletteetat

Appel	Rôle
interface.etat{projet=, fichier= nom=}	Canevas de l'éditeur d'états : zones à l'échelle, glisser-déposer, poignées, repères, hauteur des zones. fichier : etat_<nom>.prg à rouvrir ; nom : état neuf. surselection : sélection ; surchangement : état modifié.
interface.paletteetat(etat)	Palette : zones et éléments à glisser sur l'état, ou à cliquer.
e:nom() · e:code()	Nom de l'état · texte de etat_<nom>.prg (fonction ETAT_<NOM>).
e:codeexemple()	Code Lexis+ qui remplit l'état (rapport.debut, r.ajoute... avec les variables de chaque zone).

Appel	Rôle
e:noeuds()	L'état, ses zones et leurs éléments : liste de {uid, nom, genre, chemin, niveau, conteneur}.
e:selection() · e:selectionner([uid])	uid de la sélection · sélectionne (sans argument : l'état).
e:element()	{uid, nom, genre, titre, aide} de la sélection (genre : etat, zone, texte, rectangle, ligne, image).
e:proprietes()	Propriétés de la sélection : liste de {nom, genre, valeur, choix, cible} (cible geom : position, taille, hauteur en mm).
e:fixerpropriete(nom, valeur, [cible])	Modifie la sélection ; nul si c'est fait, sinon le message d'erreur.
e:renommer(uid, nom, projet)	Renomme l'état, une zone ou un élément : {message, ancien} (ancien : ancien nom de l'état, dont le fichier est à supprimer).
e:supprimer()	Supprime la sélection.
e:monter() · e:descendre()	Zone plus haut / plus bas ; élément dessiné plus tôt / plus tard.
e:nouvellezone([genre])	Ajoute une zone (zone, entete ou pied).
e:zoom([facteur])	Multiplie l'échelle ; renvoie le zoom en %.
e:aperçu(chemin, [projet])	Écrit un PDF d'aperçu (chaque zone trois fois) : {chemin, erreurs}.

Aides-mémoires

Chaque concepteur sur une page : les gestes, les fichiers, les propriétés et les instructions, avec leurs enchaînements. À imprimer et à garder sous la main.

Lexis+



Aide-mémoire · Formulaires

31

contrôles dans la palette

Dessiner une fenêtre, régler ses contrôles, relier leurs événements à vos procédures.

Voir aussi : [Interface graphique \(guide, ch. 13\)](#)

[États PDF](#)

ENCHAÎNEMENTS

Conception > Nouveau formulaire... → poser les contrôles → propriétés → événements : + nouvelle procédure... → Câbler le code

FORMCLIENT_OUVRIIR() → interface.executer() → formclient.nom.valeur

Modèle type

```
clients = {}  
FORMCLIENT_OUVRIIR()           && form_client.prg  
interface.executer()  
? chaine(#clients) + " client(s)"  
  
procedure CLIENT_ENREGISTRER(ev)  
  locale f = formclient  
  locale c = {nom = f.nom.valeur}  
  c.ville = f.ville:element(f.ville:selection())  
  inserer(clients, c)  
  f.etat.libelle = "Enregistré"  
finprocedure
```

Gestes du concepteur

glisser	de la palette : poser ; sur le dessin : déplacer
clic	palette : ajouter au conteneur sélectionné
poignées	redimensionner (la fenêtre aussi)
Suppr · Échap	supprimer · sélectionner le parent
flèches · Maj	absolue : 1 px · 10 px
Entrée	appliquer un nouveau nom
repères	bords, centres, « = btn1 », 1/4 ... tout

Fichiers et noms

form_<nom>.prg
le formulaire, réécrit à chaque modification

FORM<NOM>_OUVRIR()
ouvre la fenêtre, renvoie ses contrôles

form<nom>.<contrôle>
variable globale : formclient.nom

&& ---- formulaire « nom » : début
bloc des procédures : gardez les marques

<CONTRÔLE>_<ÉVÈNEMENT>(ev)
nom d'une nouvelle procédure (ENREGISTRER_CLIC)

wdlexis_palette.json
vos contrôles : kind, base, ctor "F(\$libelle)"

Dispositions

vertical · horizontal	empilés · côte à côte
grille	en tableau ; colonnes = 2
flux	à la suite, avec retour
bordure	ancrage haut, bas, gauche, droite, centre
pile · absolue	un seul visible · x, y

Propriétés et événements utiles

libelle · texte	étiquette, bouton · saisie
options · elements	cases, radio · combo, liste : a;b;c
etat	actif, grise, lecture
poids · ancrage	place libre · bordure
surclic	tous les contrôles
surchangement	saisie, cases, radio, curseur...
surselection	combo, liste, tableau, arbre...
surouverture	fenêtre : focus, valeurs initiales

Pièges à éviter

- ▲ Ne modifiez pas form_<nom>.prg : il est réécrit par le concepteur.
- ▲ Une procédure d'événement a un paramètre : procedure NOM(ev).
- ▲ Une procédure associée puis supprimée : corrigez l'événement dans l'inspecteur.
- ▲ Le code après interface.executer() ne s'exécute qu'à la fermeture.



Aide-mémoire · États PDF

Dessiner des zones au millimètre, les remplir avec le paquet rapport, obtenir un PDF paginé.

Voir aussi : [Fichiers & formats \(guide\)](#) [Formulaires](#)

7
instructions du paquet rapport

ENCHAÎNEMENTS

rapport.debut("facture.pdf") → r.entete("entete", {...}) → r.pied("pied", {...}) → r.ajoute("ligne", {...}) × n → r.fin() → {chemin du PDF}

Conception > Nouvel état... → Aperçu PDF → Insérer le code

Modèle type

```
r = rapport.debut("facture.pdf") && etat_facture
r.entete("entete", {numero = "2026-0142"})
r.pied("pied", {societe = "Kerlann"})
r.ajoute("colonnes", {})
r.entete("suite", {numero = "2026-0142"})
pour chaque a dans articles
  r.ajoute("ligne", {designation = a.nom, ;
    prix = a.prix})
suivant
r.ajoute("total", {ttc = total})
? r.fin(), r.nombrepages()
```

Gestes de l'éditeur

glisser · clic	poser · ajouter à la zone sélectionnée
bord bas	hauteur de la zone
flèches · Maj	0,5 mm · 5 mm
Échap	zone, puis état
repères	bords et centres ; sinon pas de 0,5 mm

rapport

rapport.debut(pdf, [etat])
commence ; etat_<etat>.prg ou nom du PDF

r.ajoute(zone, {variables})
zone à la suite ; page suivante si pleine

r.entete(zone, {variables})
page en cours si vide, sinon suivantes ; nul : aucun

r.pied(zone, {variables})
page en cours et suivantes ; nul : aucun

r.fin()
{page}, {nombre de page}, écrit le PDF, rend le chemin

r.nombrepages() · rapport.etats()
pages · noms des états du projet

Variables

{client}	donnée par r.ajoute ; vide si absente
{prix:2}	1 234,50
{page} / {nombre de page}	calculées par r.fin()
{date} · {heure}	25/10/2026 · 14:05

Zones et éléments

entete · pied	haut et bas de chaque page
autre nom	ajoutée par r.ajoute, autant que nécessaire
extensible	grandit avec ses textes
sautavant	commence une page
texte	police, taille, gras, alignement, retour
rectangle · ligne	bordure, fond, rayon · pointillé
image	PNG, JPEG ; ajuster, étirer, centre
état	A4...Legal, portrait/paysage, marges

Pièges à éviter

- ▲ Une zone ne se coupe pas : gardez les zones de détail basses.
- ▲ Un montant sans format s'affiche « 4.9 » : écrivez {prix:2}.
- ▲ Une variable mal orthographiée imprime un texte vide.
- ▲ r.entete après une zone ne vaut qu'à partir de la page suivante.